2020

Environment Search Planning Subject to High Robot Localization Uncertainty

Jared Joseph Beard
West Virginia University, jbeard6@mix.wvu.edu

Environment Search Planning Subject to High Robot Localization Uncertainty

Jared J. Beard

Thesis submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Master of Science in
Mechanical Engineering

Yu Gu, Ph.D., Chair
Jason N. Gross, Ph.D.
Natalia A. Schmid, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2020

Keywords: Decision Making under Uncertainty, Robotic Search, Vehicle Routing

Abstract

Environment Search Planning Subject to High Robot Localization Uncertainty

Jared J. Beard

As robots find applications in more complex roles, ranging from search and rescue to healthcare and services, they must be robust to greater levels of localization uncertainty and uncertainty about their environments. Without consideration for such uncertainties, robots will not be able to compensate accordingly, potentially leading to mission failure or injury to bystanders. This work addresses the task of searching a 2D area while reducing localization uncertainty. Wherein, the environment provides low uncertainty pose updates from beacons with a short range, covering only part of the environment. Otherwise the robot localizes using dead reckoning, relying on wheel encoder and yaw rate information from a gyroscope. As such, outside of the regions with position updates, there will be unconstrained localization error growth over time. The work contributes a Belief Markov Decision Process formulation for solving the search problem and evaluates the performance using Partially Observable Monte Carlo Planning (POMCP). Additionally, the work contributes an approximate Markov Decision Process formulation and reduced complexity state representation. The approximate problem is evaluated using value iteration. To provide a baseline, the Google OR-Tools package is used to solve the travelling salesman problem (TSP). Results are verified by simulating a differential drive robot in the Gazebo simulation environment. POMCP results indicate planning can be tuned to prioritize constraining uncertainty at the cost of increasing path length. The MDP formulation provides consistently lower uncertainty with minimal increases in path length over the TSP solution. Both formulations show improved coverage outcomes.

Contents

List of Figures

List of Tables

Acknowledgments

There are many to whom I owe a great personal debt for their support and the opportunities over the years. Chief among them, my committee chair, **Dr. Yu Gu**, who never fails to challenge my perspective and understanding on various issues. His patience and dedication to not only further the education of the members of the Interactive Robotics Lab and the growth of the robotics community is truly noteworthy. I would also like to extend my gratitude to **Dr. Jason Gross** and **Dr. Natalia Schmid** who provide thought-provoking discussions and have integral to my growth as a research.

Furthermore, I owe thanks the members of the Interactive Robotics Laboratory and the WVU Navigation Laboratory, with whom it has been a pleasure to work and grow alongside as researchers. In particular, I'd like to acknowledge **Nicholas Ohi** and **Jared Strader** with whom I often exchange ideas and have helped influence this work. Additionally, without the efforts of **Dr. Konstantinos Sierros**, I might not have developed my interest in research or had many of the opportunities early in my academic career.

Lastly, I have my family, in particular my parents **Jared and Angela Beard**, without whom I could not have grown into the person I am today. Sadly, this to say nothing of the countless friends and teachers who have shaped my growth and attitude over the years. Nonetheless I appreciate the opportunities I have had to spend time with them and the great deal with which I have learned from them.

This research is funded in part by an academic grant from the National Geospatial-Intelligence Agency, (Award No. # HM0476-18-1-2000, Project Title: Autonomous Navigation of UAV/UGV Teams in Underground Tunnels). Approved for public release, 20-698. This research was also made possible in part by NASA West Virginia Space Grant Consortium, Training Grant # NNX15AI01H.

Contributing Papers

This work builds upon related works completed during the course of the author's Masters studies. They are included here

- J. Gross, M. De Petrillo, **J. J. Beard** , H. Nichols, T. Swiger, R. Watson, C. Kirk, C. Kilic, J. Hikes, E. Upton, D. Ross, M. Russell, Y. Gu, C. Griffin. Field-testing of a uav-ugv team for gnss-denied navigation in subterranean environments. In Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019), pages 2112–2124, sep 2019. doi: doi.org/10.33012/2019.16912.
- M. De Petrillo, J. N. Gross, **J.J. Beard** , Y. Gu. "Exploration Planning of a UAV/UGV Team with Localization Uncertainty in a Subterranean Environment". IEEE Aerospace and Electronic Systems Magazine. (Submitted: Jun 2020)

1 Introduction

Imagine for a moment waking up after a long night out and realizing you have lost your glasses. Not only are you not sure where your glasses are, but it may not even be clear how you got here or where 'here' is for that matter. This is a significant problem for robots as they enter more human friendly environments. Where once a robot was expected to navigate a sterile environment having the highest quality sensors and very clear directions, our mechatronic counterparts now find themselves in cluttered environments with cost competitive components and tasked with increasingly vague objectives. The focus of this work therefore is in the realm of robotic path planning for search problems and errant localization sources.

This chapter is dedicated to describing the multi-objective search problem, motivations for investigation and the contributions of this work.

1.1 Motivation and Impact

Today, robots are active members in many industries: finding applications in scientific surveys [69], emergency relief [35], and capital maintenance [81], among others. As robots enter mainstream society, they are subject to increasingly unstructured environments. Whereas a lab or warehouse robot may have simple, clearly defined workspaces and top-shelf equipment, real-world applications elicit the need for robustness to complicated circumstances and ambiguities. This results in difficulties achieving a consistent fix on the robot's location. If at any moment the robot does not know where it is, then if can not know where it needs to go. Consequently, inability to converge on an accurate location estimate may result in mission failure. Overcoming localization uncertainty is a significant challenge in robotics (particularly for aerial robots). While hovering, unmanned aerial vehicles (UAV) are subject to physical and measurement drift which skew estimates. Without some ground truth to help constrain uncertainty in pose estimates, then, aerial vehicles can lose track of their location. Unmanned ground vehicles (UGV) are also subject to drift in their estimates. They however benefit from being fixed to the ground. As such, UGV can maintain a position with relative ease. Doing so allows UGV to maintain or improve their state estimate when remaining in place.

Moreover, recent interest in solving search problems is evident from the rise of research projects such as the DARPA Subterranean Challenge [18] and the NASA Space Robotics Competition 2 [78]. In both cases, the intent of search is to find useful resources, be it for emergency relief as in the DARPA Challenge or for mining applications as in the NASA Competition. These are both scenarios where human intervention will be strictly limited due to the extreme constraints of the challenges' respective environments. By sending in robots, not only can this work be done more efficiently but it can help limit human exposure to dangers. Now that NASA has committed to return to the moon by 2024 to begin developing a lunar base through the Artemis program, this is ever more important to the future of human endeavours [79]. In the case of the NASA challenge, the robots are required to search a large area. As they traverse, if the localization error grows significantly, there is considerable risk of searching areas repeatedly wasting precious time and energy. In the case of the DARPA challenge, high quality maps are necessary to report the locations of search items. Quality mapping is heavily reliant on good localization; distortions in a map could result in emergency crew getting lost or delayed care for victims.

1.2 Problem Statement

The objective of this work is to develop a path planner for a robot to search a 2D environment. Here, the robot is differential drive, having wheels fixed on parallel axes and using the velocity differential between wheels for turning. The robot will know its exact initial location, and will have a prior map of the region to be searched. At the end of traverse, the robot is expected to return to its starting location. As the robot traverses its environment, it will take location measurements through a mixture of globally fixed sources with bounded error and limited range, as well as, local sources with unbounded error growth. In this case, the global sources are beacons from which the robot is guaranteed to get a relative position estimate with negligible error when it is in range; local sources would be an inertial measurement unit and wheel encoders–both with significant noise. When only the local sources are used, the state estimate drifts as a result of integrating the noise. The specific parameters of these measurement sources are defined in Ch. 4. Given the environment is static, the planner is expected to compute a near-optimal path prior to search. Optimality criteria are set as a function of the energy cost of search and uncertainty in localization. The planner will be verified in simulation. The expectation of this work is that it will lay the groundwork for multi-robot search (wherein the beacons become mobile) and real-time 3D search.

As an important note, optimizers are described using terminology of searching a solution or state space. There is degeneracy of nomenclature here as the purpose of this work is to search for an object. It is the hope of the author the intended subject will be clear from the context. In cases where both topics are discussed in proximity, however, efforts will be made to draw a distinction. This will primarily be treated by describing the problem as search and the approaches to optimization as searching the solution or state space.

1.3 Contributions

Within the realm of robotic search and planning under uncertainty, one finds little overlap in the literature, as will be demonstrated in Ch. 3. The shift from robots in industrial and laboratory setting to real world environments, as well as, the utility of search to object retrieval tasks, indicate the need to balance these objectives. To this end the work presented here contributes the following:

- two formulations for the multi-objective search problem:

  - A Belief Markov decision process (BMDP) formulation, solved using the Partially Observable Monte Carlo Planning (POMCP) method. In doing so, later work could extend to planning online (as the robot is completing its mission).
  - A Markov decision process (MDP) approximation to the BMDP, solved using value iteration (VI). The objective here, being exploration of reasonable approximations.

- a reduced complexity state representation for the search MDP, neglecting the path history. By treating the states as independent of history, the problem complexity could be made a function of nodes, as opposed to path length. It should be noted, that while the states for both the BMDP and MDP only consider the set of visited nodes, the application of these in their respective solvers is where the benefit comes into play. VI can effectively neglect the history, however, the design of POMCP is such that it searches the state-action tree as diverging paths. Furthermore, explicit consideration of the BMDP problem relies on the history of observations to evaluate belief states, the curse of history.

In making the described contributions, this work hopes to lay the groundwork for more general multi-objective search planning tasks. This includes online planning in 3D with consideration for multiple robots, obstacles, exploration, prior information, among others. Furthermore, by exploring state space representation in the MDP, this approach could then be extended to the partially observable Markov decision process (POMDP). Given that have an intractably large number of states and must be solved using approximate methods, simplified complexity can greatly improve the ability of these methods to effectively search the solution space.

1.4 Problem Formulation

The problem presented is a Partially Observable Markov Decision Process. Here it has been simplified as a BMDP for the sake of application. The process is defined according to 4-tuple: B set of belief states b, A set of actions a, T(b' | b, a) transition probability function, and $\mathcal{R}$(s, a) reward function. Note the ' indicates a state reached from some state-action pair and that s is the true underlying state. The area of search has been discretized into a graph G = (V, E) wherein V represents the set of viable robot positions and E the viable motions. The belief states b are defined as the set $\{x, N, P\}$. Here, x is the robot pose, N is the set of visited nodes and P is the position error

covariance. For two vertices u,v $\in$ V, A := {uv $\mid \forall$ uv $\in$ E}. The reward $\mathcal{R}$ is defined as

$$\mathcal{R}(s,a) := \begin{cases} \beta(c,d,P) + a & \forall v \in N \iff v \in V \,\&\, v = v_{initial} \\ \beta(c,d,P) + \rho & v \notin N \\ \beta(c,d,P) & v \in N \end{cases} \tag{1.1}$$

Herein, a is the reward for visiting all nodes, ρ is the cost for not visiting a new node. β is the cost of traversing to the edge as function of baseline energy expenditure c, distance d, and P. β is described as

$$\beta(c,d,b) = c + w_1 * d + w_2 * (1 - n_{visited}/n)^2 * tr(P). \tag{1.2}$$

The terms w_1 and w_2 are scaling parameters. For the additional scaling term applied to $tr(P)$, $n_{visited}$ indicates the number of nodes visited and $n = |N|$. This is used to deweight the belief estimate near the end of search in lieu of tracking time and imposing an explicit constraint. In applying the baseline energy cost, the intent is to motivate the robot not to remain in place. Application of the distance term is used to bias the solution towards taking shorter paths. Lastly, consideration of the error covariance is intended to favor low uncertainty paths.

To approximate this formulation as a Markov Decision Process the set of belief states is replaced with S the set of states. States s $\in$ S are defined as s:= {u, N, t}. Here, u is the current vertex and t indicates the number of timesteps since receiving a beacon update. These timesteps are then used to approximate the $tr(P)$ as

$$tr(P) \approx e^{(\frac{d}{d_{min}} t)}. \tag{1.3}$$

Here d_{min} indicated the minimum arc length. This approximation was used to model the worst case exponential error growth for the 2D model. Scaling the time steps by the relative distance accounted for longer paths resulting in nominally greater error growth than shorter ones. While exploring cost functions, making β a function proportional to t and the cube of t were also considered. These, however, did not produce desirable behaviors in the paths and so were not investigated in detail. The square of t seemed to offer similar behavior, though this was also not investigated in detail.

2

Structure of Robotic Path Planning Problems

The task of robotic path planning involves determining the set of spatial states a robot must pass through to arrive at some state goal. Take as example the scenarios of navigating a robot to a charging station or use of a manipulator to pick up a block. Herein path planning is defined specifically as deciding the set of linear positions a robot should transition between to achieve some goal. Similar topics in robotics include motion planning which considers full poses (both angular and linear positions) and trajectory planning which factors in dynamics and control actions. In planning, obstacles present a unique danger in that they can cause mission failure or otherwise damage a robot. Obstacle avoidance may be conducted at a planning level or more reactively at a lower level. Path planning may exist as one goal among others or as a means to achieving some other objective. Under these circumstances a higher level mission planner is tasked with breaking the task into ac-

tionable steps. This is the case in robots which are expected to transport goods or safely traverse a region to find a survivor. While motion, trajectory, and mission planning play a key role in robotics they are not central to this work and will therefore be only discussed as necessary.

To investigate path planners, it is important to understand the context of the bigger robotic system: the robot and its environment. The environment consists of features which can interact with the robot (*e.g.*, terrain, obstacles, light, other agents). The robot will consist of a set of sensors used to collect information on the environment and actuators allow the robot to interact with the environment. From a sensory perspective, the field of view and quality of a sensor impacts the ability of a robot to identify an object in its environment or ability to ascertain a reliable state estimate. Similarly the quality of an actuator will impact how consistently a robot can perform a desired behavior. The interactive components of a robot do not exist in a vacuum, however; they are tied together by the body of the robot. This body has some form which defines sensor and actuator placement, encoding advantages for specific tasks. Manipulators for example will be good at grasping tasks, while a rover is expected to conduct traverse missions. Another example lie in the compliance of the body itself. Whereas rigid bodies allow for higher precision and repeatability of motion, soft bodies can be safer to work around [55]. This argument persists when we consider drivetrain, be it omnidirectional (capable of moving in any direction), differential, pedal or otherwise. Such characteristics impose requirements on planners. They must consider how reliable an output can be achieved and the constraints on motion, then compensate for these effects. Furthermore, the form of the environment plays a significant role in deciding the characteristics a planner should have. Terrains that yield easily and can cause robot to slip, impacting localization or in the case of pedal robots lead to falling. Consequently, a planner in this scenario should balance risk and avoid particularly dangerous regions. Alternatively, some regions fail to provide adequate sensory data for a given sensor suite (*e.g.*, trying to use a camera in the dark).

Within the context of a system, it becomes clear how robots fit within the broader framework of decision making and decision processes. Decision making occurs when some agent selects an action to perform based on information about both its and the environmental states [57]. In a decision process, an agent makes multiple decisions in succession to interact with its environment and other agents. A planner, as with any decision maker seeks to achieve some objective while maximizing the reward or minimizing the cost of doing so. Some factors include such details as distance travelled, information gained, uncertainty in position, and safety of a path.

The rest of this chapter is dedicated to formulating the problem of path planning in a sound theoretical framework. This begins with a discussion on what constitutes a decision process and how one applies such processes to make decisions. Because decision making relies on information about the robot and environmental states, an understanding of the means with which such states are represented is necessary. As mentioned above, information about the state must be gathered through sensors. Sensors in the real world are noisy and subject to constraints on application. Therefore a section detailing these constraints and means to overcome them through the application of sensor fusion is included as well.

2.1 Decision processes

As a robot interacts with its environment, it receives sensory input and provides output through its actuators and body. Concurrently, the environment is undergoing a set of changes due to external factors, reacting to input from the robot, and supplying information to sensors through various physical means. These phenomena occur in a continuous spacetime and at any given moment will be defined by a state. More clearly, a state $s \in S$ (the state space) consists of the information used to effectively describe a given scenario. For example, consider a point moving along a line. Its position, velocity, and acceleration would be considered sufficient information to describe the motion of the point. The point effectively exists in a one dimensional environment. This state will be affected by some action $a \in A$ (the action space) which has some probability $T(s'|s, a)$ of transitioning to the desired state. In the example of our point this would be considered the option of accelerating in one direction along its line or doing nothing. To interpret its state, the agent takes as sensory input described as an observation $o \in \Omega$ (the observation set). Such an observation is dependent directly on the true state of the system. As there is noise in real sensors, the conditional probability for receiving an observation $O(\{s_i, s_{i-1}, \ldots, s_0\}, \{a_i, a_{i-1}, \ldots, a_0\}, \{o_i, o_{i-1}, \ldots, o_0\})$ should also be considered. Depending on the context this will be assumed to be 1. If we assume that along the line our point received a certain intensity of light which varied by position, one can see how regions with similar intensity may be mistaken for others. As such looking to the past helps resolve these ambiguities. Every state transition is associated with some reward $\mathcal{R}(s, a)$; which establish the optimality criteria against which decisions are made. These rewards may then be discounted by some factor when the rewards further in the future are be deemed less valuable. Returning to the

example of the point. Perhaps it must get to a specific coordinate on the line, receiving a very large reward for arriving, but incurring a cost associated with the time and energy spent getting there.

The various classes of decision processes make assumptions regarding the observability of states, understanding of the underlying model and rewards, as well as importance of time history. Perhaps one of the most widely applied assumptions is that of the Markov property. The Markov property is defined as the characteristic of independence between future and past states given the current state [65]. Processes relying on this assumption are referred to as Markov decision processes (MDP). Additionally, MDPs are characterized by the assumption that the states are fully observable. That is, the robot has all necessary information to know its state with certainty. As such, we can neglect O as all observations will reflect the current state. What makes MDPs attractive for use is the dramatically simplified structure that come with breaking time dependence. Given motion of macroscale bodies obeys the Markov property, having a good localization source is all that is necessary to enjoy the benefits of planning as an MDP. Lastly, in the special case where transition probabilities are precisely one, MDPs become deterministic finite automata or state machines. At this point they cease to be random processes. This holds for the cases in robotics where sensation, actuation, and control capabilities can ensure every move is deliberate.

In the more general case of non-Markov decision processes, where the Markov property is broken, significant dependence on the history of states is exhibited. The boundary here comes down to the formulation or nature of the problem itself. A poignant example is the case of the diffusion of a substance across some media [97]. Formulating this problem from the perspective of the state in the media at any given time lends dependence to the direction a given particle came from. Given information regarding the state of all particles or any incident particles, however, we can reformulate the problem as one directly dependent on motion. This is to say nothing of the obvious difficulty in doing so–such a scenario is meant for explanation purposes.

Suppose on the other hand that the Markov property does hold, but the noise in sensors is significant enough to effect planning. This problem is referred to as a partially observable Markov decision process (POMDP) and is the framework through which this work most directly applies. Attempting to solve POMDPs exactly becomes intractable as it requires computation of not only all possible states, but all observation probability distributions over those states. That said, they are applicable to a wide variety of problems, including robotics, marketing, database queries, structural inspection, search and rescue, and target identification, among others [16]. Given the uncertainty

in localization and hence state inherent in robotics path planning, this is an especially powerful framework for solving this class of problems. Because of the complexity, it is helpful to simplify the problem to one of the belief state Markov decision process (BMDP). Here a belief state is one wherein states are considered with the uncertainty or belief of being in each state. This belief state is here referred to as $b \in B$ (the belief space). The idea behind the BMDP is that the belief achieved by reaching a state through some set of actions and observations should be similar whenever they occur [72]. One can then do away with consideration of all possible conditional distributions for receiving some observation, thereby considerably reducing the solution space.

Lastly, considering there is ambiguity regarding the reward for making a given action. This may be a result of uncertainty in transitions of the greater environmental states, such as motion of external agents [57]. Under these conditions the problem belongs to a class known as multi-arm bandits. As insinuated by the name, this is much like playing the slot machines because the problem involves trying an action several times to uncover about the underlying reward. The process of learning the state-action value function Q is hence referred to as Q-Learning. Such methods often rely heavily on reinforcement learning techniques given the lack of an existing model [99]. This class of problems fits within the framework of POMDPs, though given a reasonable estimate of Q for the underlying MDP, some solutions may opt to neglect partial observability to good effect [68].

2.2 Representation of state space

The planning space will rarely match the continuous spacetime of the real world. Planning through continuous time is often not a consideration for a variety of reasons. Chief among them, is that computers are inherently bad at dealing with solving continuous time processes unless they can be formulated as differential equations or otherwise fit into a matrix to be solved in one-shot. Furthermore given digital computer-based robots operate on bit strings and can only evaluate data at discrete rates, in many cases there would be insufficient information to induce a meaningful change in action at every moment in time. Spatial state representations on the other hand can be and are sometimes represented continuously. Given the computational requirements, however, many planners opt to represent spatial states in a discretized manner through a grid or graph based representation.

The state of a robot and its environment goes beyond where and when it is. There is need to consider factors such as states of various joints, locations of obstacles, the belief about these states, regions of interest, prior information, *etc.* Representation of a robot's pose in space or its joint positions may be described as the configuration. Viable joint configurations and body poses are known as the free space, while information regarding environmental factors as the obstacle space. A planner reaches some objective by planning through this configuration space as opposed to the physical one [70]. Such a representation is helpful in dealing with robotic planning for things like manipulators where planning through joint angles may be more intuitive than planning through explicit linear positions in space. This does have the drawback that obstacles and other spatial information are more difficult to represent in these transformed spaces. Taking this concept a step further, some planners choose to operate in the belief space which considers the belief for an underlying state. Additional factors such as the life of a battery remaining or the set of locations which have been visited further expand the state space. Even if the only mention of the battery life is in the cost function, one should understand that the cost is implied by a change between states from some action. Understanding these assumptions and when they are being made can make a significant difference in the representation selected for a state space–and consequently how difficult the problem is to solve. Furthermore, the state space involves more than the physical space of the robot, it also considers the environment and may consider more abstract states necessary to achieve a goal.

2.2.1 State Space Reduction

When the size of the state space is very large, searching the entire solution space becomes very computationally expensive or intractable. In some cases this can be overcome by reframing the problem; consider in the search problem tracking the set of visited states as opposed to the entire history of the robot path. In a complete graph for example this would result in a complexity of 2^n as opposed to k^m assuming we visit every node once. Here n is the number of nodes in a given k-regular graph (where all nodes have k edges) and m is the planning depth. Alternatively, one can apply a receding horizon approach (to plan within a certain time frame) or implement hierarchical solvers [81]. That said, it is often helpful to approximate the state space itself. In this case, solvers attempt to search a representative or otherwise promising subset of the state space. A quirk of path planning, however, is the number of states in the configuration space strongly impacts the num-

ber of other states such as the belief states or the regions of search in a foraging problem–wherein robots search for and transport objects to a specific location. As such, a key method of state space reduction is by approximating the spatial states. The caveat being that one also need to consider the suitability of trajectories for a given robot. Generally this approximation is accomplished through a grid or graphical method.

Formulating a space into a grid involves tessellating it, often with squares or cubes. It has benefits of ease of use from a programming standpoint given that grid cells fit into and can be indexed by arrays. As an example, the wavefront algorithm fills cells with the value indicating their distance from some objective or a flag indicating an obstacle [106]. This representation also applies to probability distributions [25] and environmental state information [40] [60]. Under such circumstances when information is encoded into the grid for use in planning, it is termed an occupancy grid, referring to the obstacles or other information occupying given cells. The downside to using a grid is use memory inefficiently. This is due in large part to the uniform nature of a grid and fixed scale. To some degree this has been overcome by adaptive grid decomposition which nests grid cells to provide more representation in information rich regions [82]. This concept has been extended to 3D grid cells (voxels) through the octrees used by the Octomap package [43]. While not strictly a grid, cellular decomposition is common in coverage problems where convex polygons are easier to plan over. The Morse-based cellular decomposition sweeps through a region with a line orthogonal to the sweep direction. When the line is subdivided by obstacles (or joined as they are passed) the method identifies a cellular region for planning [2].

Graphs may also be used to represent state spaces. Graphs consist of a set of points known as nodes or vertices connected by a set of edges or arcs. In the context of planning, edges will typically have some cost associated with them. Optionally the vertices may also have some value or information associated with them. A grid may be represented as a special case for a graph wherein all nodes are regularly spaced and connected within a radius equal to the distance of the nearest neighbor. Graphs may be projected outward in regular patterns as with grids and lattice methods (wherein a subset of possible actions are projected from initial coordinate to form a tree) [45]. Alternatively, random graphs are built by sampling a set of nodes from the environment then connecting the edges in some prescribed manner. Subsequent collision checks at each stage ensures the graph does not intersect obstacles. The distribution of samples and connectivity of the graph in turn control the complexity of the planning problem and how well it approximates the real sce-

nario. Samples may be generated randomly according to some prior distribution or be biased in the vicinity of interesting features such as obstacles [32]. Doing so ensures sufficient coverage in areas of interest. Alternatively, controls on dispersion may be applied to ensure no region is too sparse. Low discrepancy methods (*e.g.*, Hammersley, Halton, lattices methods–differing from those above which project specific template of motions) go a step further ensuring that points are sufficiently dense and do not fall along some axis [63]. These features help to ensure that graphs permit paths around obstacles and do not become disconnected. An interesting example of planner sampling is that used by the Randomly-Exploring Random Trees planner (RRT) which builds its graph in the direction of samples (constrained to some maximum distance) as opposed to using them directly in the graph [62]. When connecting graphs some methods are to connect the k-nearest neighbors or the k-nearest components to ensure short connections and prevent isolated subgraphs, respectively [63]. In the case of visibility graphs, features on obstacles such as corners are identified as nodes and connected to those unobstructed by obstacles [71]. This increases risks of collisions but tends to offer the most direct routes. Alternatively one may opt to connect all nodes within a certain radius, as with grids.

2.3 State Estimation

State estimation is integral to robotic path planning. Whether a planner operates closed loop or plans the necessary path beforehand, a robot cannot hope to reliably and safely reach a state unless it has a reasonable assessment of its current state. State estimation occurs at two levels, that of sensation and that of perception. The role of sensory equipment is to take measurements of various environmental factors. To name a few, this includes information about nearby resources such as the concentration of a gas, motion information, or the relative location between two objects. In this context, the primary concern is the relative motion of a robot to the environment and other agents for the purpose of localization. On their own, sensor data does not offer much in regard to utility. The role of perception then is to form this data into a cohesive assessment of the robot and/or the environment, as well as the belief in this assessment.

2.3.1 Sensors

This subsection is concerned with sensors primarily used for the task of robotic localization. While it is true that other sensors such as gas sensors and light sensors can, through creative application, be utilized for this purpose, this section is presented to help understand the general implications of sensor behaviors on the task of localization. It is not intended to be a comprehensive review of sensor technologies.

Inertial measurement units (IMU) are devices which track the motion of a body by measuring its acceleration, rate of rotation, and sometimes the local magnetic field. It therefore contains an accelerometer, gyroscope, and magnetometer for each spatial direction in which it is rated. IMU's take measurements at about 1 - 50 Hz though some special cases may be several times faster [3]. IMU's are subject to measurement bias which may be a function of time, and random noise. When integrated over time state estimates based on IMU tend to drift. Users must consider the gravity of vector for Earth in accelerometer data and, for missions of significant duration, Earth's rotation for gyroscope data. Assuming a stationary agent, accelerometer data may also be used to supply pitch and roll state information based on the gravity vector. Magnetometers are primarily used for bearing measurements given the relative constancy of Earth's magnetic field; this shields them to some degree from measurement drift in the heading estimate. Application should be careful of scenarios where there are magnetic features as these may introduce erroneous heading measurements.

Wheels may be outfit with encoders. Encoders are devices which detect flashes of light or magnetic field changes to determine relative motion of a disk or ring. Based on this rotational measurement and knowledge of the radius of a wheel, linear distance can be estimated. In the case of optical encoders, a disk has slots removed at regular intervals in one or more concentric rings. A light and light sensor are placed to either side of this disk. As the wheel and disk rotate the light will be seen to flash on and off. Each flash then corresponds to some angular displacement which is thus translated to linear motion by calculating the arc length the outer face of the wheel has rotated. Encoders may be subject to backlash used through a gear train, as such it is ideal that they be placed on the shaft for which the measurement is required [11]. It should be noted that higher resolution optical encoders are more likely to miss counts, particularly at higher rates. This can to some degree be limited using absolute, quadrature, or multichannel encoding strategies. Magnetic Encoders on the other hand measure the magnetic flux induced by magnets on a spinning disk, relying on the

Hall effect. As such, measurement using magnetic encoders is best applied to measuring angular velocities.

Cameras are devices which measure the wavelength of light and intensity received at a sensor array to reproduce a projection of the light emitted by the environment. These sensors are used to identify objects from visual cues (assuming optical camera) or measure distances. Depending on the sensor technology of the camera, it will measure light in different spectra (*e.g.*, visible, ultraviolet, and infrared). The lens of a camera will affect its field of view; though a vast majority are highly directional, there are wide angle lenses such as fisheye lenses which offer an approximately hemispherical view. Such wide angle lenses, however, result in image distortion at the edges, which requires compensation to provide accurate measurements [88]. Cameras making use of a single sensor are referred to as monocular. An object detected in a monocular camera can be localized in two angular directions relative to the camera. Extension to two or more sensors offers a stereo camera. These have the advantage of parallax for use estimating the pose of features in the image. Application of such measurements, robots can resolve their motion in a process called visual odometry. There are scenarios when an object of known size is observed by a camera as in the case of ARuco tags. In identifying this object, the relative pose can be computed using the dimensions of the tag and a monocular camera [77]. Both visual odometry and ARuco tags are subject to constrained localization error so long as the tag or a set of features remain in view. Over time, however, using multiple sets of features or multiple tags with space interleaved, the error grows unbounded. An example is the application of visual odometry to teach-and-repeat methods, which produces locally consistent measurements that are globally inconsistent [27]. Visual odometry is also susceptible to failure in homogeneous environments, where features may be few or nonexistent.

A broad array of technologies are dedicated specifically to the task of ranging. Among these are laser range finders (LRF), light detection and ranging (LiDAR), depth cameras, ultrasonic range finders. LRF and and LiDAR, as the names imply, use light to measure distance by applying the speed of light and time of flight of the signal. LRF is typically a single beam oriented in one direction. LiDAR on the otherhand may employ one or more beams to take measurements in two or three dimensions. Depth cameras may similarly rely on time of flight for projected light though there are applications which rely on structured light approaches [61]. Application of such technologies produces a point cloud or depth image to indicate relative pose measurements which provide useful updates for mapping [41]. Furthermore, given the number of measurement updates low un-

certainty velocity measurements can be achieved. That said, at high velocities, technology such as LiDAR are subject to dilation. Light based sensors are also susceptible to dust clouds and nonreflective surfaces. Ultrasonic ranging devices and SONAR can be applied in scenarios where such issues arise.

Global navigation satellite systems (GNSS) similarly produce range information relative to a satellite constellation. By using the time of flight of transmissions and known positions of satellites, the process of trilateration can be applied to evaluate a 3D position estimate. GNSS offer bounded error growth and sub-meter level accuracy when fused with other sensors [74]. GNSS does suffer drawbacks such as multipath interference, can only operate above ground, and risks jamming or other human sourced manipulation. In particular, GNSS cannot be applied to subterranean environments and underwater as the signals cannot be transmitted in these cases.

2.3.2 Perception

Perception gives sensory data meaning through context. For example, while it is on the camera to "see" an ARuco tag or object, the perceptive system is what deciphers the tag or object as being in a certain position within the image. In the case of localization, a variety of measurements are integrated to provide an estimate of the robot's pose. This section will focus on some common sensory modes which are fused, as well as the means of fusing such data.

Dead reckoning is the application of heading and distance sensors to estimate displacement [100]. This may include configurations such as use of wheel odometry, inertial odometry, visual odometry. Dead reckoning, while it can be performed at relatively high rates is subject to unconstrained error growth which is problematic for applications which require high precision, like mapping. To overcome this, it helps to have globally consistent measurement sources such as GNSS from which localization error will be constrained. Fusion of such technologies helps overcome the drawbacks posed by others. For example, inertial odometry can fill in the gaps between GNSS updates and provide more precise measurements, while GNSS constrains the error growth. In cases where such measurement updates are not present, it is usually to the user's benefit to integrate multiple information sources regardless. This lead to dead reckoning navigation techniques such as visual inertial odometry [66]. Regardless of whether one must rely on dead reckoning or can source external information, integrating this data requires a state estimation technique to fuse the data.

State estimation is the process of using observed data to provide a best guess as to the underlying state. There are a variety of techniques such as factor graph [46] and linear least squares estimators [92]. This discussion will focus on Bayes type filtering methods which are common in literature. Filters can be classified as parametric or nonparametric. The difference relies in how the probability distributions over states are represented, which comes down assumptions about the measurements and what posterior distributions can be expected. Parametric filters assume that a distribution can be represented with some number of parameters which define the shape of the distribution and that errors in measurements will be similarly drawn from such distributions. Of particular interest are Gaussian filters, which assume both Gaussian noise models and hence a Gaussian state estimate. Often nonparametric problems are be approximated as parametric, and the error for doing so it minimized by optimizing the bias-variance tradeoff [26]. In the application of nonparametric filters, the expectation is that sensor errors may result in ambiguities, and hence that an estimate of states cannot be reliably retained through such approximations. For a more comprehensive discussion of filtering technqiues, their derivations, and information regarding implementation of filtering techniques, the reader is directed to [94], from which the following discussion is based.

Gaussian filters provide a unimodal estimate, which is useful when there is little ambiguity between states. Furthermore, they apply to a variety of state estimation problems given that many sensors produce Gaussian noise. The Kalman filter, likely the best studied Bayes filter, is optimal for linear Gaussian signals. For nonlinear applications, the extended Kalman filter provides an approximate solution by linearizing the state transitions [76]. Alternatively, unscented Kalman filter can be applied directly to nonlinear state models [5]. To work with the inverse of the covariance, the Information filter, is useful scenarios where quantifying information gain is important [93]. Given the underlying mathematical structure, it also offers improved efficiency over the Kalman filter when there are more measurements for state updates than state predictions.

Nonparametric filters are ideal when distributions are likely to be multimodal or when the structure of the distribution cannot be readily accounted for. Nonparametric filters approximate distributions as discrete. In the case of histogram filters, the space is approximated by discrete regions, then subsequent application of Bayes updates to each of the regions yields a posterior distribution [49]. The particle filter on the other hand relies on a set of samples called particles [96]. By randomly drawing from these particles and applying updates to the new samples, a new distribution can be generated. Practically, the number of bins or particles necessary to accurately represent a

distribution can be prohibitive to some systems.

3

Solving Robotic Path Planning Problems

Path planners operate in complicated environments as demonstrated in the previous chapter. As such, a plethora of methods have been developed to tackle planning, each with its own assumptions. These assumptions are rooted in the information available and uncertainty associated with the problem. This chapter reviews the state of the art methods applied to solving decision processes and solving planning problems specifically. Through these examples, the objective of this chapter is to ground the importance of the topics in the last chapter to the approaches employed for planning. Furthermore by highlighting the characteristics of these approaches, the discussion will show how these methods lay the groundwork for multi-objective planners such as that demonstrated in this work.

This chapter begins with a discussion of techniques applied to solving decision processes in general. Given solving decision processes involves searching a state space, the discussion will be guided by the structure of the optimization techniques employed. For the sake of brevity, this work will

assume the transition and reward model is known; as such methods, like Q-learning, for evaluating these terms will be neglected. This also neglects machine learning based approaches, such as those which use neural networks, in favor of methods with more theoretical grounding. This is followed by an account of planning methods from the perspective of their primary objectives. These include planning to reach a destination, to gather information, and with the intent to cover a region or complete a circuit. Planners will be detailed according to their benefits and the assumptions made by their solvers. At the conclusion, the reader will note that despite the variety of applications there is a clear gap between planning for search and considerations for localization uncertainty. While there are some solutions, they consider uncertainty primarily as a fallback option. Furthermore, from the broad set of literature in decision processes available, there are a breadth of techniques available for application to this problem.

3.1 Optimization in decision processes

Whatever the purpose of evaluating Markov decision processes, they are optimization problems which may or may not be constrained. As such, a broad array of optimization tools are available including dynamic programming methods. Dynamic programming methods are those which solve the problem by recursively solving more simple problems [10]. Other methods are too broad to classify under an umbrella, but they typically consist of balancing global and local search, to optimize the whole process (up to some horizon), such as evolutionary algorithms [8].

Within the set of dynamic programming based approaches there are exact [38] and approximate methods [85]–though given enough time many approximate methods will produce an exact solution. Among the exact methods are the various forms of value iteration (VI) [7] and policy iteration [44]. Value iteration seeks to optimize the Bellman equation by iteratively estimating the value of states from the value of their neighbors and reward for taking some action. Once the values have converged to stable values, a policy for each state is selected from the highest valued neighbor. Value iteration is extended to approximate POMDPs through point-based value iteration. Point based value iteration alternates between performing VI and selection of a representative set of candidate beliefs [84]. Operating on a similar principle, policy iteration, based on a current policy evaluates its value. Then based on these values it computes a new policy and iterates over these solutions [57].

The approximate methods exploit the state-action sequences involved in a decision chain to build a tree. At every node in the decision chain where a different action could be chosen or a selected action results in a different state, the tree branches. The objective of these solvers then is to search the tree as efficiently as possible to produce a near-optimal solution. Some of the earliest methods applied here were the breadth-first and depth first search algorithms; a classical example is their application to winning games [83]. Breadth first search relies on the expansion of all nodes at each iteration up to some depth. Depth first search algorithms, such as forward search, seek to explore the best action to its conclusion or to some depth at one iteration. While these solutions produce a depth optimal solution, they are inefficient as they attempt to search the entire solution space. Methods such as branch and bound seek to reduce unnecessary search by providing bounds on the state-action values to prune the search tree; this however has the worst case complexity equal forward search [64]. Alternatively sparse sampling uses generative models to randomly sample the outcomes of state-action pairs, approximating the state space and overcoming some of the complexity [52]. The Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) extends these ideas to the POMDP problem, by randomly sampling belief state outcomes and bounding the search tree [59]. Alternatively, some approaches rely on strictly stochastic means to evaluate which actions are best for a given state. Take for example the Monte Carlo Policy evaluation which develops a policy by performing a random rollout (selection of random actions or states at each node) and using a generative model of a process to search the solution space [90].

The methods presented until now rely on determinism or stochasticity to produce optimal results, but there are means which integrate the two ideas. Algorithms such as the Monte Carlo Tree Search (MCTS) balance pragmatic (actions taken directly to achieve a goal) and epistemic (actions taken to learn about the problem) behaviors to search the solution space [14]. At each iteration, available state-action pairs are expanded; this is followed by a random rollout. Because explicit simulation is time consuming and complex, the goal is that by introducing random behavior to approximate the outcome, search can be improved. The value of states-action chains are weighted by the relative frequency with which they have been tried to encourage exploration. Depending on the size of the problem then, the user can tune the amount of randomness introduced by the rollout depth and the willingness to explore solutions about which little is known. MCTS has the advantage that its complexity is a function of the number of iterations and rollout depth as opposed to the size of the search space. This approach is extended to POMDPS with the Partially Observable

Monte Carlo Planning (POMCP) algorithm by planning over belief states [89].

Other planners try to solve the problem directly. This class of solvers is quite broad in their approaches, but the general principle is quite similar. Come up with candidate solutions, evaluate their quality, use this information to create more candidates, and iterate. The core differences between these algorithms lie primarily in how they represent their information and how they utilize the information gained from an iteration to develop the new candidates. This will involve a balancing act between employing some minor changes to conduct local search and dramatic changes to the solutions to search the global space. Doing so the idea goes that any extrema can be optimized, but prevent to some degree getting caught in local optima–a common issue in local search methods such as hill climbing. Here again, the idea of pragmatic and epistemic actions comes into play. Evolutionary algorithms are a popular choice for global optimization which seek to mimic the "survival of the fittest" behavior seen in nature [8]. These algorithms produce a set of solutions at each iteration and evaluate their fitness or utility. Among these, the best solutions are kept. Then by introducing random changes, combining solutions to produce new solutions, or inducing minor variations, the solution space is searched. These methods are best used for offline purposes, as they are meant to run long enough to search as much of the space as possible. When the solution is subject to changing length, these solutions can be troublesome to implement. They do however offer a broad application for problems like deriving functional relations and parameter selection. Particle swarm optimization relies on a set of samples from the solution space [53]. Ascribing to each particle some position in the solution space and a velocity with which to move through it, their fitness will be evaluated iteratively. Depending on the value of the solutions around them, forces will be applied to the particles, guiding them through the solution space. Depending on the size of the solution space, a significant number of particles may be necessary to effectively search the solution space, making the method quite computationally intensive. These represent some of the more broadly applied. There are a variety of other algorithms such as memetic algorithms [80], and simulated annealing [56] which exist to perform global optimization as well.

3.2 Planning

The task of path planning has long existed in contexts beyond robotics. Early examples consider famous problems in mathematics such as the travelling salesman problem (TSP) where travellers

such as 19th century circuit judges and lawyers would attempt to work a set of cities in the shortest distance [6]. In the context of graph theory and the state representations described in Ch. 2, this involves visiting a set of nodes to produce the shortest circuit. One can readily imagine how this applies to a road trip today or using a robot to cover an area. This section then describes a variety of path planning problems tackled by the robotics community and the assumptions upon which they depend. Whether implicit or explicit, though, it is necessary to point out these planners optimize routes against some cost function with factors such as distance, energy, risk of failure, information gained or otherwise.

3.2.1 Planning for a Destination

The goal of traverse is generally for a robot to navigate its environment. While this could involve a set of destinations, this section is aimed at the limiting case of arriving at one specific location. As with the more general decision process framework, these planners involve some means of search through a set states, specifically spatial ones. There are however some creative approaches which solve the task indirectly. Mimetic approaches such as potential fields, generate a field with the goal acting as a sink and entities as sources. The robot then may simply follow the field to arrive at its destination [98]. This has the advantage of providing a policy for any starting destination but is subject to local minima traps without some additional perturbations. This approach also does not guarantee optimality.

In the realm of path planning, the solution space is physical space; for breadth first search planning this is akin to propagating wavefront from a given position. This has given rise to a variety of so called Fast Marching methods (FMM) which stem from the study of wavefront propagation in physics. They involve solving the Eikonal equation which describes the motion of the wavefront [101]. FMM reaches a solution by incrementing the value of occupancy grid positions as the wave moves an increasing distance from some source. When applied to path planning the robot is treated as the source. Once the wave has been propagated, application of gradient ascent yields the path plan. In the case of the Fast Marching Squared, the front is propagated from obstacles. The further from an obstacle, the faster the motion which can safely be achieved by a robot. By then solving the FMM on this velocity field, Fast Marching Squared integrates speed allowing for a time optimal solution, not just distance optimal [95]. These algorithms and a host of other Fast solutions

are described in [37]. This class of planners generally rely on a grid given its regularity. While they will find an optimal solution, wavefront planners are not very efficient as they involve search of the entire search space.

Best-first search methods apply additional knowledge about a problem to guide search towards the answer. So is the case with Dijkstra's algorithm [17]. Dijkstra's algorithm assumes you have some graph where each node has an estimated cost associated with reaching it. From some starting position, the planner searches neighboring nodes and computes the actual cost to go. If this cost is less than the estimate, the cost is updated. Selecting the lowest cost node which has not been expanded as the next to search, the algorithm iterates through the graph. This continues until the goal is reached for some cost which is less than any other available trajectory. A* extends the best-first concept by including heuristic knowledge about the proximity to the goal [39]. As such, when considering the costs, the nodes which are expanded are those with the least cost, consisting of the sum of verified or estimated cost and the projected cost to the goal. These heuristics should, however be admissible and consistent, having lesser cost than that of the overall cost and satisfying the triangle inequality, respectively. Otherwise, the search would not be guided properly. The heuristics are otherwise open to weighting as deemed necessary to guide the search. These search algorithms come with several variants. Among them, the methods appended by a * are incremental and store previous efforts to speed up replanning, as in A*. Anytime algorithms are adapted to provide a time optimal solution and can improve as it goes [67]. Bidirectional search methods root trees at both the start and goal to better guide search [73]. Backward search methods start at the goal which can aid replanning. Lite algorithms are versions of their predecessors which are easier to implement [58]. An important point to note for graphical implementations is the use of the probabilistic roadmap (PRM) as a framework for planning. This employs a two step method wherein a viable graph is constructed randomly then a solution is queried according to a search method of the user's choice, providing a powerful framework [51].

Whereas PRM plans the graph beforehand, there are planners which build the graph incrementally. Among these graphical methods, rapidly exploring random trees (RRT) and rapidly exploring random graphs (RRG) are noted here [50]. The planners randomly sample a point and then extend to the nearest node some distance in that direction. Whereas RRT will only extend one point to form a tree, RRG also extends those nearby to form a more general graph. These methods are well suited for including constraints on trajectories. Furthermore, RRT* is probabilistically

complete and will converge on an optimal solution as the number of samples in the tree grows. As with the best-first planners described above, these are classes of planners with many variants for improving the trees, replanning, using bidirectional search, among others.

These problems are easily formulated in an MDP framework since for this section we are assuming there is no localization uncertainty. Hence value iteration and policy iteration work quite well with these problems [57]. Naturally methods such as forward search can be extended to these problems. While some have applied global planners [87], they are not necessary to get good performance given the relatively small size of most monotonic planning problems.

3.2.2 Planning for Information Gathering

In addition to reaching some position, robots are often tasked with information gathering about the robot and its environment to reduce localization uncertainty, search an environment, or explore and map an environment. Such approaches fit in with the theme of active perception, wherein decision making and control are applied to the process of data acquisition [9].

To improve localization, there have been a variety of belief space planners. Algorithms such as the Rapidly Exploring Random Belief Trees [15] and the Belief Roadmap (BRM) [86] extend RRT and PRM to plan over belief states. Belief space planning suffers the curse of history, as the trajectory impacts the reward received at a given spatial state. This curse of history is the concept that the uncertainty of states is dependent on previous states, and hence that history must be considered in planning. To overcome this hindrance, the Feedback-based information Roadmap builds on BRM using control techniques to drive the robot to a specified position then use time to allow localization to converge on and constrain belief states at nodes [4]. It should be noted in this work, however the underlying assumption is that some global information source exists with which to constrain these belief states. Alternatively, given the monotonic behavior of maximum eigenvalues at belief states, Bopardikar, *et al.* [12] propose their application to bound uncertainty for belief planning problems. For planning in the continuous domain, Indelman, *et al.* [47] extend the belief state to include information about the environment. In their work, they propose solving the planning problem as a POMDP in two layers, propagating information about the global state, then propagating and optimizing the control input directly. SARSOP has been applied to the task of underwater navigation in this regime, where localization updates are rare [59]. Another approach,

the Determinized Sparse Partially Observable Tree algorithm approximates the belief tree by a set of sampled scenarios to break the curse of history [91]. The impact of this is that the planner need not consider the entire tree; the algorithm has been put to good effect in planning for autonomous vehicles.

Search and exploration problems are closely intertwined. Whereas search implies one is looking for some object in an environment, exploration is search applied to information. As such, exploration planners may seek to expand frontiers of the known environment and perform mapping functionalities. Mapping here consists not only of the spatial environment, but also characteristics and fields such as the temperature of ocean water or plankton concentrations. As is the the case in the information maximization planner of Low, *et al.* [69]. The Rapidly-exploring Information Gathering planner builds on RRG to include information gained at each node and was shown to yield promising results for mapping wireless signal strength when compared to branch and bound [42]. SARSOP, when given a highly uncertain map, was applied to the task of improving the quality of the map and localization while reaching goal [59].

Yamauchi pioneered the idea of frontier-based planning using occupancy grids for the task of exploration by identifying regions where no information has been collected [103]. Bourgault, *et al.* [13] worked to improve map quality by balancing information gain about the map with localization performance from simultaneous localization and mapping. More recent work by Papachristos *et al.* [81] uses a receding horizon approach to integrate mapping and exploration. They apply a hierarchy to solve the problem in two stages. They complete the coverage and mapping problem by building a search tree for desired coordinates, then perform belief space planning to ensure spatial consistency.

For search with no prior information, the problem amounts to one of coverage, as described in the next sub-section. There are some works, however, which apply existing knowledge to the task of search. Ye [105] considers both prior distributions on object location and viewpoint planning for perception of 3D objects in search. Wong, *et al.* [24] plan over prior information with a receding horizon approach and apply a Bayesian search filter to update the estimate of object states. The 2016 NASA Sample Return Rover Centennial Challenge winner, Cataglyphis, relied on Bayes type updates to estimate the likely locations of objects. Then based on this information search the next best viewpoint [36]. This approach considered a spatial observation model for the camera, distance, and time remaining. Alternatively, Doctor, *et al.* [20] applied particle swarm optimiza-

tion to robotic search. These search methods, however, did not demonstrate explicit consideration for minimizing uncertainty.

3.2.3 Planning Circuits and for Coverage

Most of the previously discussed planners have had objectives which require visiting relatively few states. Coverage problems involve the task of exhaustively traversing an entire region given some radius of coverage the robot can sense or act upon. Depending on the context, these can be considered a special case of the search problem. The expectation is that the coverage planner will interact with every bit of the environment, whereas search planners need not always do so. Circuit and route planning differ in that they involve returning to the starting coordinate and may sometimes only require visiting a subset of nodes or vertices. Discussion of routing problems will focus primarily on TSP, the Chinese Postman Problem (CPP), and their variants.

Early coverage planners relied on lawnmower or zigzag-like behavior to cover convex regions. This led to the need for the cellular decomposition methods, such as the Morse decomposition previously described. Several algorithms, however, rely on grid and graph based approaches. In coverage, wavefront algorithms are again useful, though instead of traversing the gradient, the objective is to traverse equipotentials, only deviating when necessary to make progress [106]. Spiral algorithms utilize grids with 2 x 2 subgrids. By traversing along an edge in the subgrid and only turning when a boundary is met, the planner can always ensure a viable route home with minimal recoverage [28]. Some methods, such as [29] consider localization uncertainty, though it is used as a fallback and not incorporated directly into the plan. Acar, *et al.* [1] though, exploit knowledge of the environment by following cell boundaries and using these to provide information about location. For a more complete review, the reader is directed to that by Galceran [30]. In this survey, he details classical methods which rely heavily on cellular decomposition to produce convex hulls for classical zigzag planners, grid and graph based methods, multi-agent methods, methods under uncertainty, optimal coverage, and 3D coverage.

The objective of TSP is to visit a number of cities at the least cost and return to the origin. Mathematically speaking this amounts to traversing all the vertices in a graph once, except the origin at the lowest edge cost. Applegate cites a number of approximate solution methods such as cutting planes from convex hulls, using linear programming solvers, and searching trees [6]. A more recent

development is the application of ant colony optimization, a global optimization technique [104]. In this case, a number of random solutions are rolled out. The best among them leave "pheromones" which guide the search of subsequent rollouts.

For the Chinese Postman problem (CPP) or route inspection problem, a person must deliver mail along a set of streets without repeatedly visiting a street [48]. This in turn amounts to traversing all the edges in a graph once and returning to a starting vertex. To solve the problem exactly requires all nodes have an even number of edges. Fortunately, the CPP is relatively easy to evaluate when edge costs are fixed. This is accomplished by generating random circuits. Then for any nodes with unused edges, simply repeat this procedure, replacing a node in the primary circuit with the new circuit. The solution of any tour will then be of the same cost. This is of course neglecting consideration for the more difficult problem of graphs with odd nodes. CPP has several variants of which many are NP-Hard; of interest are the windy postman problem and rural postman problems. The Windy postman problem has edge costs which depend on the direction of traverse. An example would be a greater expense in gas for a mail truck to drive up a street as opposed to down or the belief of a robot entering a region with low uncertainty localization updates versus leaving one. A prominent solution to this problem is the application of cutting planes [102]. The Rural postman problem requires only a subset of edges be traversed at minimal cost. Such may be the case when a robot is expected to search a few small regions for an object, but the greater area is known not to contain anything. Such was the case in the NASA Sample Return Robot Challenge [75]. Some approaches include application of the polyhedron for the graph and applying cutting plane algorithms [19] such as branch-and-cut (extends branch and bound to include cutting planes for tightening bounds) [33]. For solutions to these and other variants, the reader is directed to the following reviews [21] [22] [23].

4

Simulation Environment

To evaluate the approach presented this work employs the MATLAB and C++ programming languages. Matlab provided the benefit of rapid prototyping and easily employed graphics interface. C++ on the other hand provides significant advantages in terms of efficiency and open source packages. Of these packages, use of the Robot Operating System (ROS) not only provided a simplified interface for communication between various programs, but also access to the high fidelity Gazebo simulation environment. ROS has a broad community acceptance, offering a solid foundation for online support in debugging various issues. Gazebo offers a broad array of sensor models, evaluation of dynamics, multiple physics engines, and more [31]. Additionally, in versions directly integrated with ROS, there is functionality for handling transforms between bodies, logging through ROS interfaces, and adaptability between physical and simulated robots. The decision to forgo testing on a physical robot platform was made after careful consideration. Given the concurrence of the COVID-19 pandemic with the final results of this work, access to facilities has

been limited. While there is no substitute for such physical tests it is believed that the work herein can be suitably represented in the given simulation environment. Future work in this direction will be extended to the physical Pathfinder robot, described below, as the future situation permits.

This chapter details the simulation environment and parameters for the robot under consideration. First the available sensor models employed and their nominal operating conditions are detailed. This is provided alongside the robot and environment models. The state estimation strategy employed is described next. For all simulations be they done strictly in MATLAB/C++ or in coordination with the Gazebo, all parameters specified will be consistent unless otherwise specified.

4.1 Robot and Environment Models

To facilitate future testing in the real world, this work uses a simulated model of the Interactive Robotics Laboratory's (IRL) Pathfinder robot (Fig. 4.1.1). Pathfinder is a four-wheel differential drive robot used primarily for testing and development. As such, the robot has a significant software foundation and can be easily reconfigured to meet testing needs. Furthermore, there is an existing simulation package containing a Pathfinder model (Fig. 4.1.2) with which to utilize. The relative ease of use made Pathfinder a straightforward choice of model. The real robot is outfit with an ADIS 16495-2 IMU having 1.6°/hr drift and quadrature encoders having 47,000 pulses/m resolution [54]. Given noise levels may vary from one test to the next, velocity and heading noise were arbitrarily drawn from the respective normal distributions

$$v \sim \mathcal{N}(0, 0.1\, m/s) \tag{4.1}$$

and

$$\dot{\psi} \sim \mathcal{N}(0, 0.10). \tag{4.2}$$

Measurements were simulated at frequency of 50 Hz. Control of the rover velocity and angular velocity was accomplished with proportional controllers, having gains Kv = 1, and K$\dot{\psi}$ = 5, respectively.

Beacons were placed throughout the environment to provide a global update. The beacons are

Figure 4.1.1: Image of the IRL Pathfinder Robot, reproduced with permission from the author [54]

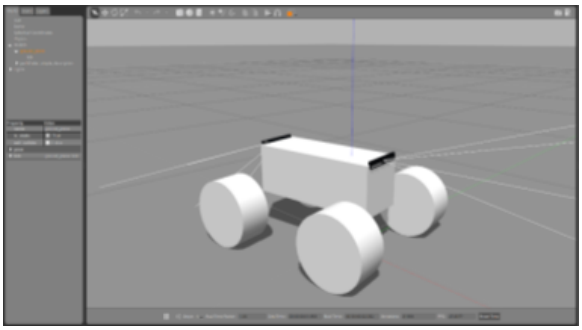

Figure 4.1.2: Simulated Pathfinder robot in the Gazebo environment

assumed to contain cameras looking in 360°and being able to communicate with the robot. The robot would then carry ARuco codes arranged in a cube atop the body. In this way the updates would be provided at any point within view of the camera–here treated as a distance of 7.5 m. The observation probability in this region was assumed to be 1. As the ARuco library can resolve a full pose, this would result in measurements of the 2D position and heading for the robot. The beacons were assumed to provide measurements at a rate of 1 Hz. The measurements were modelled as having no drift. Noise for the linear directions and heading were arbitrarily drawn from the following distributions

$$x \sim \mathcal{N}(0, 0.01\,m) \tag{4.3}$$

and

$$h \sim \mathcal{N}(0, 0.010). \tag{4.4}$$

A 2D plane was selected for the environment. As for the search environment, the graph is arranged with nodes at 15 m intervals in the x and y directions (Fig. 4.1.3). Beacons were placed at coordinates (16,16) m and (16,31) m. Connections are made for nodes within a radius of 23 m. In this way the transitions act as they would on grid with diagonal and lateral connections. The environment considered measured 45 m x 60 m and the robot was assumed to have a 7.5 m radius sweep. As such in a perfect coverage solution, there would still be uncovered area in the 4 corners. This was done as a simplifying assumption; while the nodes could have filled the bounding box, it was expected this would skew the results towards more coverage, making it harder to differentiate outcomes. As such, perfect performance was defined to have a coverage of 95.41 %. The sweep of the robot and a total coverage example can be seen in Figure 4.1.4.

4.2 State Estimation

For the purposes of this work, the relevant state of the robot is its pose. Given this is a 2D motion problem, the pose consists of an two linear dimensional coordinates x and y, and one angular dimension the heading ψ. This yields a state vector

$$\hat{x} = [x,\ y,\ \psi]^T. \tag{4.5}$$

While the model exists in 3D, motion in the z direction, pitch, and roll should be negligible given the low speeds and flat environment. Because the noise models are Gaussian, an Extended Kalman Filter was selected for state estimation in this problem. The prediction step utilizes velocity v and yaw rate $\dot{\psi}$ from the gyroscope. This results in a predicted state for time step k

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + \begin{bmatrix} vcos(\dot{\psi}) \\ vsin(\dot{\psi}) \\ \dot{\psi} \end{bmatrix} dt_{k|k-1} \tag{4.6}$$

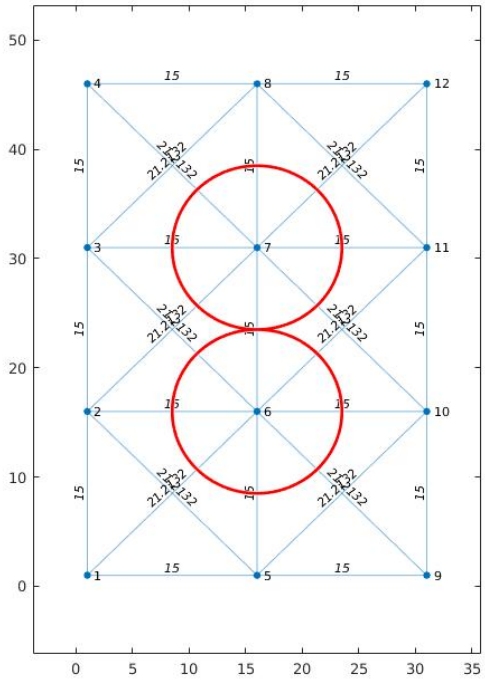

Figure 4.1.3: Arrangement of beacons (red circles) and graph construction.

where $dt_{k|k-1}$ is the time difference between steps k and k-1. The Jacobian for the predicted state F is therefore

$$F_k = \begin{bmatrix} 1 & 0 & -v\sin(\dot{\psi})dt_{k|k-1} \\ 0 & 1 & v\cos(\dot{\psi})dt_{k|k-1} \\ 0 & 0 & dt_{k|k-1} \end{bmatrix}. \tag{4.7}$$

When available the the beacon update supplies a full state measurement

$$z_k = [x,\ y,\ \psi]^T_{meas}. \tag{4.8}$$

Consequently the observation is

$$h = \hat{x}_{k|k-1}. \tag{4.9}$$

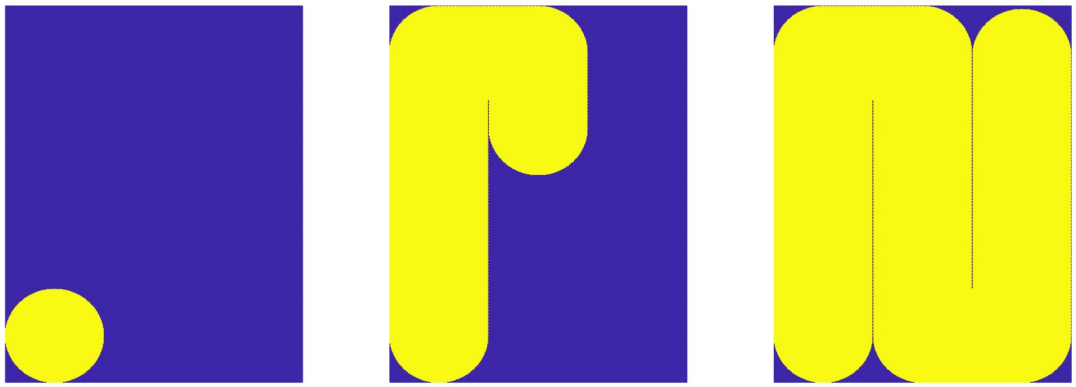

Figure 4.1.4: (left) Sweep of robot in environment; (middle) partial coverage example, (right) full coverage.

Given the state equations are nonlinear, the prediction covariance Q_k is estimated as

$$Q_k = \begin{bmatrix} \frac{v}{\sqrt{(2)}} & 0 & 0 \\ 0 & \frac{v}{\sqrt{(2)}} & 0 \\ 0 & 0 & \omega \end{bmatrix}. \tag{4.10}$$

The update measurement noise R_k however, can be known exactly and is hence set as

$$R_k = \begin{bmatrix} x & 0 & 0 \\ 0 & x & 0 \\ 0 & 0 & h \end{bmatrix}. \tag{4.11}$$

With this information, we can then compute the filter's estimate for the k^{th} time step. The filter is derived and described in more detail in *Probabilistic Robotics* [94]. First the filter predicts the covariance estimate

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k. \tag{4.12}$$

Assuming there is an update, the Kalman gain is computed. Because the observation h is linear, its

Jacobian is an identity matrix. The Kalman gain here is thus simplified to

$$K_k = \frac{P_{k|k-1}}{P_{k|k-1} + R_k} \tag{4.13}$$

and the residual is computed as

$$r_k = z_k - h. \tag{4.14}$$

Lastly, these are used to update the state and covariance estimates as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k r_k \tag{4.15}$$

and

$$P_{k|k} = (I - K_k) P_{k|k-1}, \tag{4.16}$$

respectively. In the last equation, note again, that the term for Jacobian of h simplifies out.

5 Algorithms

The work presented here considers multiple approaches to search planning. The differing characteristics in the two formulations impacted how the structure of the problem could be exploited and hence the means to arrive at a solution. Given the solution space is too large to evaluate exactly for the BMDP, efficient search of the solution space was necessary. Furthermore to support a later shift from offline to online performance, the solution should be independent of the size of the search tree. POMCP offered the means to control the time invested in this search and the ability to balance exploitative and exploratory search. For the MDP formulation, problems could more easily be defined with a finite set of states. As a result, VI was selected to evaluate an exact solution. The proposed formulations were compared to the TSP solution as evaluated with Google's OR-tools software package. The TSP solution served as a baseline for optimal coverage planning with no consideration for uncertainty. In doing so, the relative performance benefits of the approaches employed could be characterized. The remainder of the chapter details the algo-

rithms employed and implementation details for each of the approaches.

5.1 Value Iteration

To evaluate a problem using VI iteration necessitates listing of all states to be searched. By exploiting this list and the formulation, the problem was greatly simplified. To see how, one must first remember that the states are defined by the current vertex, set of nodes visited, and the number of timesteps since receiving a beacon update. In the search problem, one is generally concerned with all nodes being visited as a path. By treating the set of nodes visited as unordered, however, one can enjoy a significant reduction in complexity. In the case of the graph used for this problem, the search space of paths is bounded by $mn^2 3^k$ and $mn^2 8^k$ states. Here k is the length of the path, m the maximum number of timesteps since receiving a beacon update, and n the number of nodes. One n term is for the number of vertices the robot could be in, while the other is for each starting vertex. The powers of 3 and 8 define the minimum and maximum number of incident edges for the nodes, representing the possible number of action to some depth. By not considering the order in which these nodes are visited, however, resulted in a complexity of $mn^2 2^n$. Here 2^n accounts for the number of combinations of visited nodes. This breaks the dependence on search horizon, as well. For twelve nodes, the solution took ~3 min to run using an Intel i7 processor. Given the complexity of the problem, this therefore requires several hours to compute a solution for problems with more than 16 nodes. For all states, the transition probability is defined as 90 % chance of reaching the desired node and 10 % chance of staying in place. In practice there is some nonzero chance of transitioning to a different node. As this formulation does not consider path history, there was no clear way to allocate this appropriately beyond setting the transitions to be equal, so they were neglected.

In applying VI to the problem, the following parameters were used in evaluating the reward structure. The parameters were arrived at by incrementally evaluating solutions. Qualitatively this amounted to comparing how short the paths were and how effectively they would reduce uncertainty from a human intuition. For example, spacing the beacon updates relatively evenly throughout the paths would limit the exponential growth of uncertainty. As defined in Chapter 1, a indicates the reward for reaching the goal, ρ is the cost for not visiting a new node, c is the baseline energy expenditure cost, w_1 is the weight for distance related costs, and w_2 is the weight for belief

related costs.

$$a = 10,000, \tag{5.1}$$

$$\rho = 0, \tag{5.2}$$

$$c = -25 \tag{5.3}$$

$$w_1 = -50, \textit{and} \tag{5.4}$$

$$w_2 = -1. \tag{5.5}$$

The value iteration algorithm involves successive approximations of a state's value. This is done by estimation from the greatest sum of the reward for available actions and expectation of the values of the resulting states. Performing this estimate iteratively until convergence, then exactly solves the Bellman equation–resulting in the optimal values for all states. Value iteration is completed by selecting as the policy at each state, the action which results in the greatest value. This algorithm is detailed in [57] and shown in Algorithm 1. Here no discount is used, so the term is neglected. V indicates the value of state.

Algorithm 1: Value Iteration

Result: V_k

$k \longleftarrow 0$

$V(s) \longleftarrow 0 \qquad \forall s \in S$

while $\neg converged$ **do**

  $V_{k+1}(s) \longleftarrow \max_a [R(s,a) + \sum_{s'} T(s'|s,a) V_k(s')] \qquad \forall s \in S$

  $k \longleftarrow k + 1$

end

VI was applied offline (that is evaluating the policies before use) in a MATLAB environment. Spatial states considered only the vertices of the graph. Kinematics were therefore not modelled,

with motion described by only the beginning and ending positions. Whereas VI should involve generating policies for all states, the implementation considered each starting coordinate as a separate problem (though they were included for the complexity described above). As such, VI was run once for each starting coordinate to get a set of policies. This was done to simplify the code base; running the problem offline and for relatively few states meant this would not cause undue burden. Following evaluation of the policies, they were simulated in the C++/Gazebo environment.

5.2 Partially Observable Monte Carlo Planning

Because evaluation of all states in the BMDP would be intractable, an approximate solver was used. POMCP offered several benefits over alternative methods given that it is designed specifically for solving POMDPs online (evaluation of solutions at the time of the mission). That the search was bounded by the number of iterations the user selects implied it could be put to use for anytime planning, supporting replanning, which would be desirable when this work is later extended. Furthermore, POMCP applies a generative model from the underlying system permitting the planner to treat the problem with the BMDP formulation.

In applying POMCP to the problem, the following parameters were used in evaluating the reward structure. As defined in Chapter 1, a indicates the reward for reaching the goal, ρ is the cost for not visiting a new node, c is the baseline energy expenditure cost, w_1 is the weight for distance related costs, and w_2 is the weight for belief related costs. The reader will note the difference in parameters as compared to VI. This difference stems from the difference in problem structure. Given that VI was applied to an MDP which not only had differing assumptions, but also reward and transition models, as well. From a fundamental level, this amounts to solving two different problems. Attempts were made, however, to use the same parameters. The POMCP solutions, under such circumstances did not appropriately balance the objectives. This resulted in solutions favoring one objective or which were otherwise nonsensical. Furthermore, the BMDP solution appeared to have a broader range of parameters in which it could effectively balance the multiple objectives. To evaluate this range in more detail multiple uncertainty weights were considered.

$$a = 1,000,000, \tag{5.6}$$

$$\rho = -500, \tag{5.7}$$

$$c = -250 \tag{5.8}$$

$$_1 = -1, \textit{and} \tag{5.9}$$

$$w_2 = -5. \tag{5.10}$$

The POMCP algorithm is based on three functions: search, simulate, and rollout. The search function randomly selects from the set of belief states. It then performs a simulation on this state. These operations are repeated until timeout; at timeout, the function returns the best action. The simulation algorithm, if a state has not yet been extended, extends the tree for all actions from the state. It then calls the rollout function to estimate the original state's value. Otherwise, the function selects an action based on the neighbor with the greatest sum of the reward and an exploration term scaled by some factor c. Using this action, it applies a generative model G to evaluate the reward received by taking the desired action, then simulates one layer deeper. The reward and value from the simulation are used to update the reward for the state. Next the state-action tree is updated with the value of the state and number of visits to the state; the belief state is appended to the set. Lastly, the rollout function up to some depth recursively follows a random policy, using a generative model to propagate the states. The reward obtained is then the sum of the recursively estimated rewards.

For this work the POMCP algorithm is modified such that termination of the search depends on the number of iterations as opposed to a time limit. Additionally, the simulation step only extends a single node, as opposed to doing so recursively to some depth. POMCP is detailed in [89]. The search, simulate, and random rollout algorithms are presented in Algorithm 2, 3, and 4, respectively with the described modifications. Additionally, when the random rollout concludes, a default policy defined by the TSP solution continues selecting actions until the goal is achieved to better approximate the reward; see Algorithm 5. For the following algorithms h refers to the state-action history or path of the robot, N indicates the number of times a given path has been searched, V is the value of a state, B indicates the belief state, and π indicates a policy, whether random or

default.

Algorithm 2: Search(h)

n $\longleftarrow$ 0

Result: $\text{argmax}_b\, V(hb)$

while $n < n_{max}$ **do**

  $s \sim B(h)$

  $Simulate(s, h)$

  $n \longleftarrow n + 1$

end

Algorithm 3: Simulate(s,h)

Result: R

if $h \notin tree$ **then**

  $tree(h,a) \longleftarrow (N_{init}(ha), V_{init}(ha))\forall a \in A$

  $R \longleftarrow RandomRollout(s, h, 0)$

else

  $a \longleftarrow \text{argmax}_b[V(h, b) + c\sqrt{\frac{logN(h)}{N(hb)}}]$

  $(s',o,r) \sim \mathcal{G}(s, a)$

  $R \longleftarrow r$

  $B(h) \longleftarrow B(h) \cup s$

  $N(h) \longleftarrow N(h) + 1$

  $N(ha) \longleftarrow N(ha) + 1$

  $V(ha) \longleftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$

end

Algorithm 4: RandomRollout(s,h,depth)

Result: R

if $depth < depth_{max}$ **then**

  $a \sim \pi_{random}(h)$

  $(s', o, r) \sim \mathcal{G}(s, a)$

  $depth \longleftarrow depth + 1$

  $R \longleftarrow r + RandomRollout(s', h, depth)$

else

  $R \longleftarrow r + DefaultRollout(s, h)$

end

Algorithm 5: DefaultRollout(s,h)

Result: R

while *goal* $\neg$*reached* **do**

  $a \sim \pi_{default}(h)$

  $(s', o, r) \sim \mathcal{G}(s, a)$

  $R \longleftarrow r + DefaultRollout(s', h)$

end

In application of this problem, the full 2D robot pose was used to define the state, along with the belief, and set of visited nodes. The belief was represented by a multivariate Gaussian distribution and propagated using an extended Kalman filter. The generative model was based on the change in pose between the starting and ending coordinates with measurements having noise simulated as defined in Ch. 4. In determining updates, there was the option to utilize the maximum likelihood estimate obtained from the simulated motion or to randomly draw a sample from the belief. Randomly drawing from the distribution showed no apparent preference for receiving belief updates. Consequently, it was assumed that drawing from the distribution showed an unrealistic expectation of inadvertently receiving belief updates. For this reason, the maximum likelihood estimate was used to determine if the robot received a beacon update. POMCP was applied offline and so the goal was to evaluate the theoretically best performance by near-exhaustively searching the relatively small set of possible paths. In application, POMCP was found to select a policy for 100,000 iterations of the search step with no rollout in ~1 min when run offline. To determine these search

parameters, Monte-Carlo trials were performed to evaluate the expected reward as a function of the number of search iterations and the depth of rollout. For each case, 100 trials were performed. From Figures 5.2.1 and 5.2.2 the expected reward converged around between 10,000 to 50,000 iterations. The decision to use 100,000 was made in light of the the relatively small number of trials. Furthermore, given the planning was done offline, the relatively short time was not a constraint. In regard to the rollout, depths greater than zero showed decreased performance in the solution (Fig. 5.2.3). This is believed to be a side effect of the relatively small size of the problem and the additional default rollout. Given these can search the solution space relatively thoroughly compared to larger problems, the additional randomness likely biased the search in favor of worse solutions.

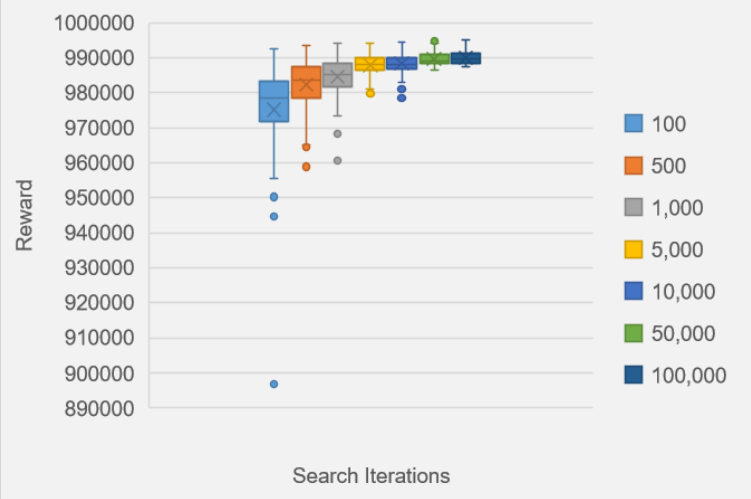

Figure 5.2.1: Convergence of POMCP expected reward as a function of the number of simulation iterations for $w_2 = -2$.

5.3 Travelling Salesman Planner

To evaluate the TSP solution, the Google OR-tools package was implemented. OR-tools performs combinatorial optimization for constraint programming, linear and mixed-integer programming, and vehicle routing tasks, as well as other graph related algorithms [34]. The problem was framed

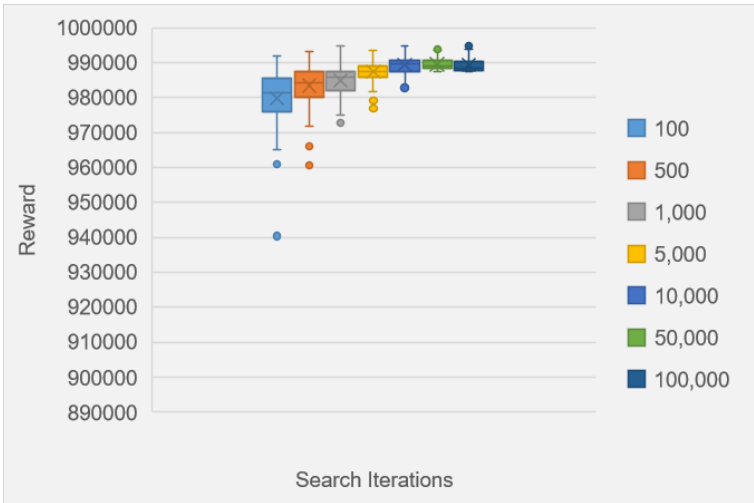

Figure 5.2.2: Convergence of POMCP expected reward as a function of the number of simulation iterations for w_2 = -5.

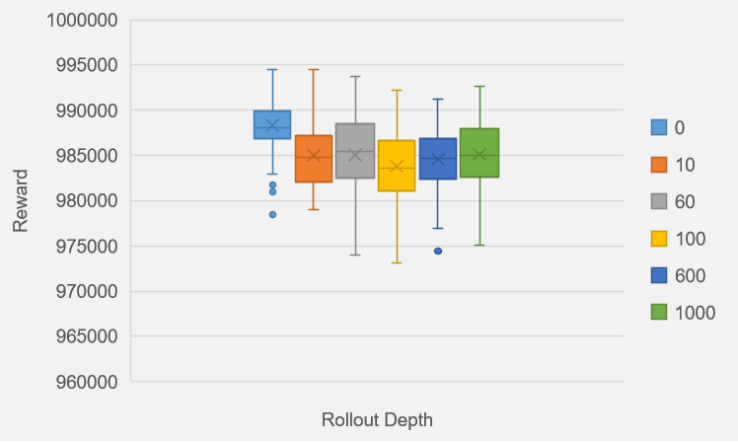

Figure 5.2.3: Convergence of POMCP expected reward as a function of the rollout depth for w_2 = -2.

as an adjacency matrix with adjacency values indicating the distance between two nodes. In the instances where no connections existed, the edge costs were set to values much larger than the distance between nodes. This is because OR-tools does not support explicit declaration of nonexistent edges for TSP. The program was instructed to plan for a single vehicle. As edge costs were not bidirectional there was no need to evaluate solutions from multiple starting coordinates; they

should all be effectively the same or flipped about an axis of symmetry. For the *routingModel* class, default parameters were used. This however led to suboptimal solutions so the depth first search algorithm was used in its stead. The TSP solution was evaluated offline before testing performance in the C++/Gazebo environment.

6 Results

Evaluation of the methods employed consisted of Monte Carlo trials characterizing the error, coverage, and path length of each planner's solutions. These results are presented alongside maps demonstrating the trajectories of the examples; therein red lines indicate the state estimate and green lines the true path. Additionally, blue circles outline the regions with beacon updates and blue arrows mark the starting coordinate, pointing towards the first stop on the path. Because the control was sufficient that the robot closely followed its estimated state, explicit plots of these policies are not included as they can be inferred from the trajectory figures.

For the Monte-Carlo trials, 24 tests were considered for each method; more trials would have been preferable, however, the length of a simulation was prohibitive. To collect general performance data, these were distributed such that each node was the starting point for two trials. In the case of the TSP, the policy was applied once in each direction from the starting coordinates. Given the TSP costs are the same regardless of direction, this offered a more complete picture of the per-

formance in the full problem. The MDP/VI solution generated one policy for each node which was tested twice. The offline BMDP/POMCP was used to generate two policies from each starting coordinate and uncertainty weight to better reflect the stochastic nature of the planner. Before discussing the outcome of all the trials, the following discussion evaluates the performance of some individual trajectories with respect to the character of the planned path. For the purposes of discussion the MDP/VI solution will be referred to simply as VI and the BMDP/POMCP solutions as POMCP.

From the trajectories of each approaches, one can see the emphasis towards the specified cost functions. This serves as a verification that the algorithms were operating as directed and the formulations achieved their respective goals. For the TSP, wherein the cost was based solely on distance, the planner clearly avoids taking diagonal edges (Fig. 6.0.1). This does, however result in deteriorated performance when the planner starts at or near nodes with updates. By receiving updates sequentially, they are unable to limit explosive error growth in the worst cases. Conversely, when the updates are further along the path from the starting coordinate, performance is somewhat better as the error is not permitted to compound over exceptionally long periods (Fig. 6.0.2).

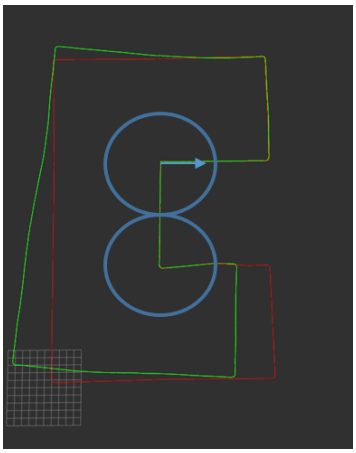

Figure 6.0.1: TSP trajectory starting in region with belief update.

For the VI to compensate for the uncertainty terms, some diagonal motions were necessary. This was so that it distributed updates more evenly throughout the trajectory, as was expected (Fig. 6.0.3). Revisiting nodes was neglected in all cases, except those where starting in a node which

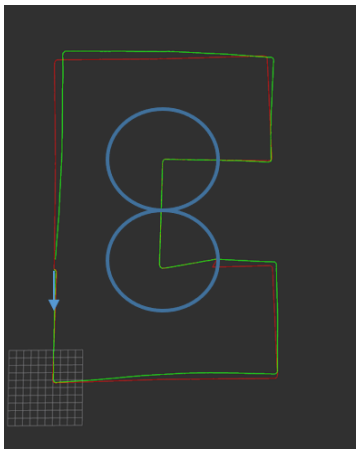

Figure 6.0.2: TSP solution with trajectory receiving update approximately halfway through.

received an update (Fig. 6.0.4).

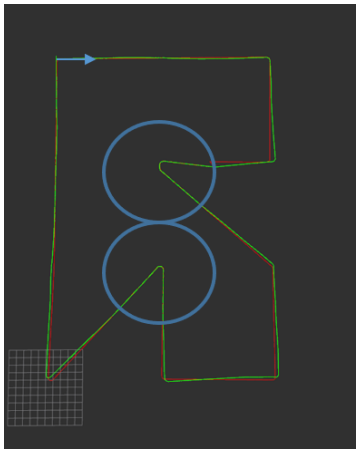

Figure 6.0.3: Prototypical VI iteration trajectory.

For POMCP with $w_2 = -5$ (POMCP$_5$), the consideration for uncertainty was much greater a priority than path length. As such, many paths were seen to weave in and out of regions with updates several times (Fig. 6.0.5). That said, there were some solutions that behaved similar to VI, though they were in the minority. Furthermore, the deweighting of updates near the end of the trajectory

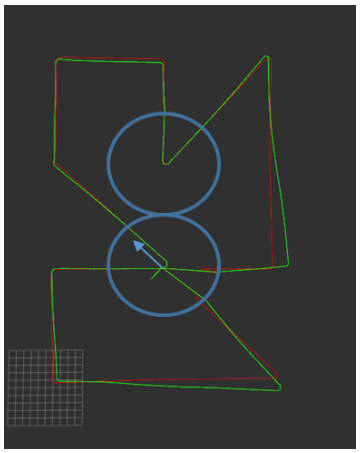

Figure 6.0.4: VI solution starting in a region with belief update.

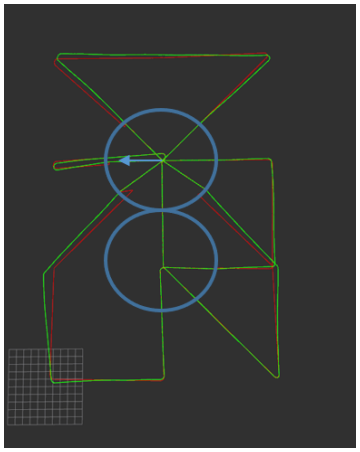

Figure 6.0.5: POMCP5 trajectory which returns for multiple belief updates.

is evident from paths which seemingly rush towards the goal near the end (Fig. 6.0.6).

In evaluating POMCP with w_2 = -2 (POMCP2), uncertainty and path length considerations were more balanced. Because of this, the paths more closely mirrored those of the VI (Fig. 6.0.7). Given the approximate nature of the planner, actions which provided no benefit did occasionally creep into the solutions (Fig. 6.0.8).

The remainder of this discussion draws from Tables 6.0.1, 6.0.2 which detail the mean and vari-

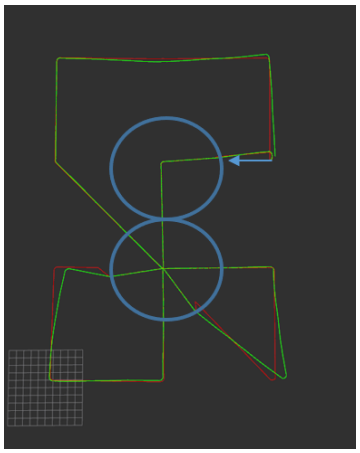

Figure 6.0.6: POMCP5 trajectory which shows less consideration for receiving updates near the end.

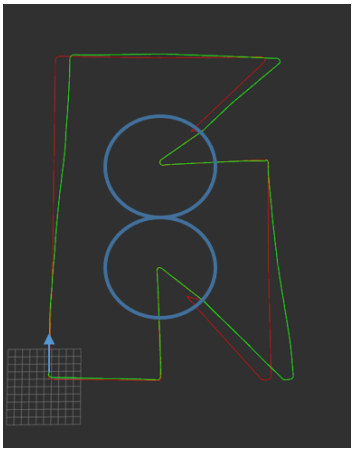

Figure 6.0.7: POMCP2 trajectory which with minor path length increase over TSP.

ance of the performance of each method in regard to their average error, maximum error, coverage, and path length. This information is presented graphically in Figures 6.0.9-6.0.12. TSP showed about 75 % of trials having less than 5 cm of average error. VI and POMCP5, however fell entirely below 4 cm. VI and POMCP5 also more than halved the average error of the TSP solution with negligible variance. POMCP2 performed approximately as well though showed greater variability in the solutions. As for their maximum errors, the TSP solution averaged about 6 m and had a variance of 10 m. This reflects the lack of consideration for constraining uncertainty. VI and both

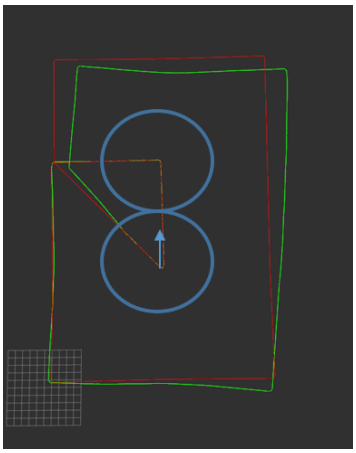

Figure 6.0.8: POMCP2 trajectory which performs unnecessary actions.

POMCP, showed a decrease of ~33 % for the mean of the maximum error with a decrease in variance of >60 %. The behavior of VI and both POMCP were within a few percent for their error. This is well within the means of having a statistical anomaly given the relatively small number of trials. That said, the POMCP5 performed slightly better in both average and maximum error in the trials. Both POMCP though were subject to greater deviation in performance than VI. Given POMCP relies heavily on randomness, and VI produces the same plan every time, this would seem to line up with expectations. It is therefore noteworthy that the relatively small variation in expected rewards resulted in considerable variability between trials.

In regard to coverage, all planners achieved greater than 90 % coverage on average and had negligible deviation from this performance. Of interest is that under these circumstances, VI and both POMCP improved the performance by ~3 %. Most of the coverage benefit seems to have come directly from the increase in uncertainty performance. Despite the extensive increases in path coverage produced by the POMCP5 solution this did not contribute substantially to a gain in coverage. Given successful coverage was 95.41 %, the VI and both POMCP solutions came considerably close to exceeding this threshold.

The minimum distance which could be travelled under perfect conditions to achieve the coverage was 180 m; TSP closely reflected this behavior. Given that TSP should provide the minimum distance which could be travelled, it is within expectation that VI and both POMCP resulted in increases when considering localization uncertainty. VI and POMCP2 saw an increase of ~10 %

	TSP	VI	POMCP5	POMCP2
Average Error (m)	0.04	0.02	0.02	0.02
Maximum Error (m)	6.05	3.81	3.67	4.22
Coverage (%)	90.50	93.18	93.86	92.53
Path Length (m)	180.29	198.23	227.28	195.14

Table 6.0.1: Mean of performance for Monte-Carlo trials

	TSP	VI	POMCP5	POMCP2
Average Error (m)	5E-04	5E-05	8E-05	2E-04
Maximum Error (m)	9.92	3.47	3.57	3.33
Coverage (%)	0.10	0.03	0.03	0.07
Path Length (m)	8.57	66.90	584.50	83.22

Table 6.0.2: Variance of performace for Monte-Carlo trials

with relatively tight clustering. The only outliers for VI were those cases wherein the starting coordinate received an observation. POMCP5, however, saw the mean path length increase by ~32 % over the TSP results. Furthermore, given the variation in behavior from solutions, this caused dramatic differences in distance travelled by the set of POMCP5 solutions. While the increase in path length falls in line with significant attempts to improve localization performance, the notable variance for both POMCP again reflects the approximate nature of this algorithm.

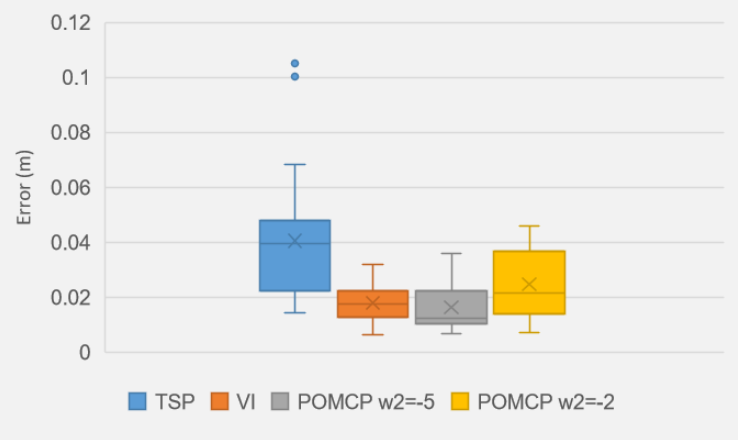

Figure 6.0.9: Average Error from trajectories.

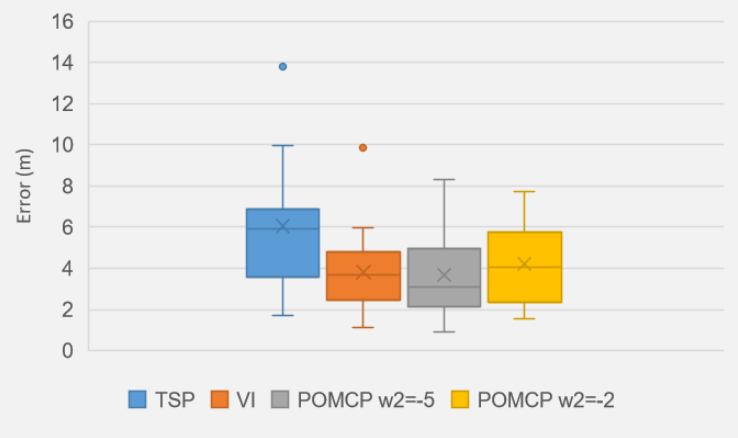

Figure 6.0.10: Maximum Error from trajectories.

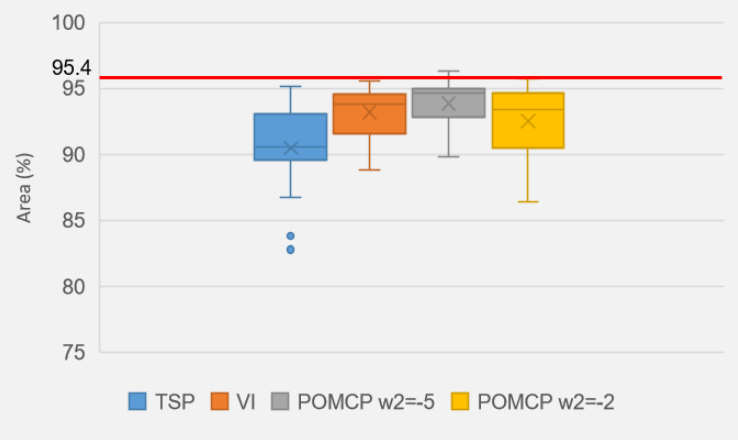

Figure 6.0.11: Area Coverage for trajectories.

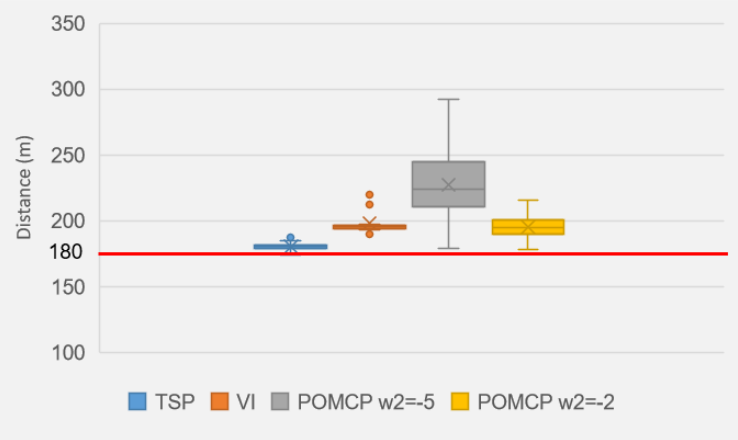

Figure 6.0.12: Path length for trajectories.

7

Concluding Remarks

The work presented here demonstrated the successful application of decision process formulations for the multi-objective search problem with high robot localization uncertainty. Performance was evaluated using VI and POMCP to produce paths for the MDP and BMDP formulations, respectively. Both showed significant decreases in error while improving coverage relative to the TSP solution. MDP/VI was demonstrated to do so more consistently than both BMDP/POMCP and TSP. Application of VI did have the drawback, however, of being tied directly to the complexity of the MDP. As a consequence it becomes prohibitively time consuming to evaluate problems requiring more than 16 nodes. Weighting the POMCP yielded similar uncertainty performance for the weights considered. As the magnitude of the uncertainty weight increased, however, significant increases in path length were exhibited. Furthermore, BMDP/POMCP showed more variability in performance than VI. This is reflected in the incomplete search of the state-action tree. Given the success of the reduced complexity state representation, it is also worth investigating extension

to the POMDP problem. Continued work will be done to further analyze the capabilities of these methods and extend their application.

7.1 Future Work

The work presented herein represents a stepping stone on the way to a broad array of more complex problems. In particular, there is need to expand this work to larger graphs given the limit imposed by Value Iteration solutions. The application of algorithms such as MCTS and DESPOT seem to offer a reasonable means towards achieving these objectives. Along such lines, improving the efficiency of or devising better POMDP solvers is also within the scope of future work. As it relates to robotics specifically, this work can be extended to 3D search and inclusion of nonuniform prior distributions of objects to bias the search. This is to say nothing of relaxing constraints on prior information so that the robot may be required to conduct exploration as well. Such is the case in disaster scenarios such as those following earthquakes where the environment may have substantially changed from prior knowledge. In line with consideration for 3D environments, the need for multiple agents with varying capabilities is worthy of consideration. Consider a UAV/UGV team such as that in [35]. UAVs can rapidly search a region from the air. They are, however, subject to significant localization drift and short battery life. Use of the UGV to help constrain error and offer a charging station can to some degree overcome these limitations. Coordinating the motion between these agents also presents interesting challenges in timing to ensure the localization uncertainty can be reliably constrained.

A key aspect in coverage and search problems is the spatial nature of the observation or actions. A lawnmower covers some region of fixed size as it moves, and one might expect the waypoints to be separated by about the size of the mower. As these problems move into the realm of the 3D space, an observation may change size, shape, or heading depending on the motion of a robot. In application of a UAV to the problem of search, it faces the choice of flying far overhead to get an approximate idea of relevant features or flying near the ground to gather more certain information. In this case a camera has some spatial field of view of the ground and accuracy of detection–both of which are dependent on the observation height. When considering prior information about the possible locations for survivors, then it will be more efficient to adjust the height to account for this information. Factoring this information into the sampling then can reduce the state space search

for planners. Considering again this is a spatially varying observation, one can see the parallel to packing problems. Additionally, given sensors with spatial observations–like a camera with a fish-eye lens–wherein the belief in a measurement is dependent on the pose from which it is sensed, some amount of overlap between observations may be necessary to fulfil mission requirements. One is inclined to think of this as a packing problem wherein the bodies deform. A potential alternative is to frame some posterior belief as an objective. Considering a set of observations and their resulting posterior distributions, one can attempt to match this posterior belief. This problem has potential to be treated as a function or model approximation problem. Given the considerable gap, in so far as there is no explicit research to my knowledge, in optimizing coverage by factoring in the spatially varying observations, this could be a very fruitful direction for research into search tasks.

References

[1] E. U. Acar and H. Choset. Exploiting critical points to reduce positioning error for sensor-based navigation. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 4, pages 3831–3837 vol.4, 2002.

[2] Ercan U. Acar, Howie Choset, Alfred A. Rizzi, Prasad N. Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21:331 – 344, 2002.

[3] Norhafizan Ahmad, Raja Ariffin Raja Ghazilla, Nazirah M Khairi, and Vijayabaskar Kasi. Reviews on various inertial measurement unit (imu) sensor applications. *International Journal of Signal Processing Systems*, 1(2):256–262, 2013.

[4] Ali akbar Agha-mohammadi, Suman Chakravorty, and Nancy M Amato. Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33(2):268–304, 2014. doi: 10.1177/0278364913501564.

[5] Muhammad Latif Anjum, Jaehong Park, Wonsang Hwang, Hyun-il Kwon, Jong-hyeon Kim, Changhun Lee, Kwang-soo Kim, et al. Sensor data fusion using unscented kalman filter for accurate localization of mobile robots. In *ICCAS 2010*, pages 947–952. IEEE, 2010.

[6] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, USA, 2007. ISBN 0691129932.

[7] Yossi Aviv and Awi Federgruen. The value iteration method for countable state markov decision processes. *Operations research letters*, 24(5):223–234, 1999.

[8] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23, 1993.

[9] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[10] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[11] John Billingsley, Alex Ellin, and Gregor Dolsak. The design and application of rotary encoders. *Sensor Review*, 2008.

[12] Shaunak D Bopardikar, Brendan Englot, Alberto Speranzon, and Jur van den Berg. Robust belief space planning under intermittent sensing via a maximum eigenvalue-based bound. *The International Journal of Robotics Research*, 35(13):1609–1626, 2016. doi: 10.1177/0278364916653816.

[13] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 540–545 vol.1, 2002.

[14] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[15] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation*, pages 723–730, 2011.

[16] Anthony R. Cassandra. A survey of pomdp applications.

[17] Jing-Chao Chen. Dijkstra's shortest path algorithm. *Journal of Formalized Mathematics*, 15 (9):237–247, 2003.

[18] Timothy Chung. Darpa subterranean (subt) challenge. https://www.darpa.mil/program/darpa-subterranean-challenge, 2018.

[19] Angel Corberán and JM Sanchis. A polyhedral approach to the rural postman problem. *European Journal of Operational Research*, 79(1):95–114, 1994.

[20] S. Doctor, G. K. Venayagamoorthy, and V. G. Gudise. Optimal pso for collective robotic search applications. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1390–1395 Vol.2, 2004.

[21] Jack Edmonds and Ellis L Johnson. Matching, euler tours and the chinese postman. *Mathematical programming*, 5(1):88–124, 1973.

[22] Horst A Eiselt, Michel Gendreau, and Gilbert Laporte. Arc routing problems, part i: The chinese postman problem. *Operations Research*, 43(2):231–242, 1995.

[23] Horst A Eiselt, Michel Gendreau, and Gilbert Laporte. Arc routing problems, part ii: The rural postman problem. *Operations research*, 43(3):399–414, 1995.

[24] El-Mane Wong, F. Bourgault, and T. Furukawa. Multi-vehicle bayesian search for multiple lost targets. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3169–3174, 2005.

[25] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987.

[26] Scott Fortmann-Roe. Understanding the bias-variance tradeoff. *URL: http://scott. fortmann-roe. com/docs/BiasVariance. html (hämtad 2019-03-27)*, 2012.

[27] Paul Furgale and Timothy D Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010.

[28] Y. Gabriely and E. Rimon. Spiral-stc: an on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 954–960 vol.1, 2002.

[29] E. Galceran, S. Nagappa, M. Carreras, P. Ridao, and A. Palomer. Uncertainty-driven survey path planning for bathymetric mapping. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6006–6012, 2013.

[30] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, pages 1258–1276, 2013.

[31] Gazebo. What is gazebo? http://gazebosim.org/tutorials?tut=guided_b1&cat=#WhatisGazebo?, 2020.

[32] Roland Geraerts and Mark H. Overmars. *A Comparative Study of Probabilistic Roadmap Planners*, pages 43–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-45058-0. doi: 10.1007/978-3-540-45058-0_4. URL https://doi.org/10.1007/978-3-540-45058-0_4.

[33] Gianpaolo Ghiani and Gilbert Laporte. A branch-and-cut algorithm for the undirected rural postman problem. *Mathematical Programming*, 87(3):467–481, 2000.

[34] Google. About or-tools. https://developers.google.com/optimization/introduction/overview, 2020.

[35] De Petrillo Matteo Beard Jared Nichols Hayden Swiger Tommy Watson Ryan Kirk Connor Kilic Cagri Hikes Jacob Upton Emily Ross Derek Russell Matt Gu Yu Griffin Christopher Gross, Jason. Field-testing of a uav-ugv team for gnss-denied navigation in subterranean environments. In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*, pages 2112–2124, sep 2019. doi: doi.org/10.33012/2019.16912.

[36] Yu Gu, Jared Strader, Nicholas Ohi, Scott Harper, Kyle Lassak, Chizhao Yang, Lisa Kogan, Boyi Hu, Matthew Gramlich, Rahul Kavi, et al. Robot foraging: Autonomous sample return in a large outdoor environment. *IEEE Robotics & Automation Magazine*, 25(3):93–101, 2018.

[37] Javier Victorio Gómez González. *Fast Marching Methods in path and motion planning: improvements and high-level applications*. PhD thesis, Universidad Carlos III de Madrid, 2015.

[38] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

[39] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[40] Marcel Häselich, Marc Arends, Nicolai Wojke, Frank Neuhaus, and Dietrich Paulus. Probabilistic terrain classification in unstructured environments. *Robot. Auton. Syst.*, 61(10): 1051–1059, October 2013. ISSN 0921-8890. doi: 10.1016/j.robot.2012.08.002. URL https://doi.org/10.1016/j.robot.2012.08.002.

[41] Sebastian Hening, Corey A Ippolito, Kalmanje S Krishnakumar, Vahram Stepanyan, and Mircea Teodorescu. 3d lidar slam integration with gps/ins for uavs in urban gps-degraded environments. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0448. 2017.

[42] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014. doi: 10.1177/0278364914533443.

[43] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. doi: 10.1007/s10514-012-9321-0. URL http://octomap.github.com. Software available at http://octomap.github.com.

[44] Ronald A Howard. Dynamic programming and markov processes. 1960.

[45] T. Howard, M. Pivtoraiko, R. A. Knepper, and A. Kelly. Model-predictive motion planning: Several key developments for autonomous mobile robots. *IEEE Robotics Automation Magazine*, 21(1):64–73, 2014.

[46] V. Indelman, S. Williams, M. Kaess, and F. Dellaert. Factor graph based incremental smoothing in inertial navigation systems. In *2012 15th International Conference on Information Fusion*, pages 2154–2161, 2012.

[47] Vadim Indelman, Luca Carlone, and Frank Dellaert. Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments. *The International Journal of Robotics Research*, 34(7):849–882, 2015. doi: 10.1177/0278364914561102.

[48] Ellis L. Johnson Jack Edmonds. Matching, euler tours and the chinese postman. *Mathematical Programming*, 5 1:88–124, 1973.

[49] Colin W Jemmott, R Lee Culver, and NK Bose. Passive sonar target localization using a histogram filter with model-derived priors. In *2008 42nd Asilomar Conference on Signals, Systems and Computers*, pages 283–287. IEEE, 2008.

[50] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *CoRR*, abs/1005.0416, 2010. URL http://arxiv.org/abs/1005.0416.

[51] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[52] Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2-3):193–208, 2002.

[53] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[54] Cagri Kilic, Jason N. Gross, Nicholas Ohi, Ryan Watson, Jared Strader, Thomas Swiger, Scott Harper, and Yu Gu. Improved Planetary Rover Inertial Navigation and Wheel Odometry Performance through Periodic Use of Zero-Type Constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 552–559. IEEE, 2019. doi: 10.1109/IROS40897.2019.8967634.

[55] Sangbae Kim, Cecilia Laschi, and Barry Trimmer. Soft robotics: a bioinspired evolution in robotics. *Trends in biotechnology*, 31 5:287–94, 2013.

[56] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[57] Mykel J. Kochenderfer, Christopher Amato, Girish Chowdhary, Jonathan P. How, Hayley J. Davison Reynolds, Jason R. Thornton, Pedro A. Torres-Carrasquillo, N. Kemal Üre, and John Vian. *Decision Making Under Uncertainty: Theory and Application*. The MIT Press, 1st edition, 2015. ISBN 0262029251.

[58] Sven Koenig and Maxim Likhachev. D^* lite. *Aaai/iaai*, 15, 2002.

[59] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *In Proc. Robotics: Science and Systems*, 2008.

[60] S. Laible and A. Zell. Building local terrain maps using spatio-temporal classification for semantic robot localization. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4591–4597, 2014.

[61] Benjamin Langmann, Klaus Hartmann, and Otmar Loffeld. Depth camera technology comparison and performance evaluation. In *ICPRAM (2)*, pages 438–444, 2012.

[62] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[63] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, USA, 2006. ISBN 0521862051.

[64] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[65] Alberto Leon-Garcia. *Probability, Statistics, and Random Processes for Electrical Engineering*. Pearson/Prentice Hall, Upper Saddle River, NJ, third edition, 2008. ISBN 9780131471221 0131471228.

[66] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[67] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *Advances in neural information processing systems*, pages 767–774, 2004.

[68] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up, 1995.

[69] Kian Hsiang Low, John M. Dolan, and Pradeep Khosla. Information-theoretic approach to efficient adaptive path planning for mobile robotic environmental sensing. In *Proceedings of the Nineteenth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'09, page 233–240. AAAI Press, 2009. ISBN 9781577354062.

[70] Tomás Lozano-Pérez. *Spatial Planning: A Configuration Space Approach*, pages 259–271. Springer New York, New York, NY, 1990. ISBN 978-1-4613-8997-2. doi: 10.1007/978-1-4613-8997-2_20. URL https://doi.org/10.1007/978-1-4613-8997-2_20.

[71] Tomás Lozano-Pérez and Michael A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, October 1979. ISSN 0001-0782. doi: 10.1145/359156.359164. URL https://doi.org/10.1145/359156.359164.

[72] David A. McAllester and Satinder P. Singh. Approximate planning for factored pomdps using belief state simplification. *CoRR*, abs/1301.6719, 2013. URL http://arxiv.org/abs/1301.6719.

[73] Bo-bo Meng and Xiaoguang Gao. Uav path planning based on bidirectional sparse a* search algorithm. In *2010 International Conference on Intelligent Computation Technology and Automation*, volume 3, pages 1106–1109. IEEE, 2010.

[74] Enge Per Misra, Pratap. *Global Positioning System: Signals, Measurements, and Performance*. Ganga-Jamuna Press, rev. 2nd ed. edition, 2011.

[75] Monsi Roman Molly Porter. Nasa's centennial challenges: Sample return robot challenge. https://www.nasa.gov/directorates/spacetech/centennial_challenges/sample_return_robot/about.html, 2017.

[76] Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Intelligent autonomous systems 13*, pages 335–348. Springer, 2016.

[77] Rafael Munoz-Salinas. Aruco: a minimal library for augmented reality applications based on opencv. *Universidad de Córdoba*, 2012.

[78] NASA. Space robotics challenge 2. https://www.nasa.gov/content/space-robotics-challenge-2, 2019.

[79] NASA. Artemis: Humanity's return to the moon. https://www.nasa.gov/specials/artemis/, 2020.

[80] Ferrante Neri and Carlos Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[81] Mascarich Frank Khattak Shehryar Dang Tung Alexis Kostas Papachristos, Christos. Localization uncertainty-aware autonomous exploration and mapping with aerial robots using receding horizon path-planning. *Autonomous Robots*, 43(8):2131–2161, 2019. URL https://doi.org/10.1007/s10514-019-09864-1.

[82] M. Parashar and J. C. Browne. On partitioning dynamic adaptive grid hierarchies. In *Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences*, volume 1, pages 604–613 vol.1, 1996.

[83] Judea Pearl and Richard E Korf. Search techniques. *Annual Review of Computer Science*, 2 (1):451–467, 1987.

[84] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[85] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[86] Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, 2009. doi: 10.1177/0278364909341659.

[87] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. *IEEE Transactions on industrial informatics*, 9(1):132–141, 2012.

[88] Davide Scaramuzza and Katsushi Ikeuchi. Omnidirectional camera. 2014.

[89] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2164–2172. Curran Associates, Inc., 2010. URL http://papers.nips.cc/paper/4031-monte-carlo-planning-in-large-pomdps.pdf.

[90] Satinder P Singh and Richard S Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158, 1996.

[91] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1772–1780. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5189-despot-online-pomdp-planning-with-regularization.pdf.

[92] Peter JG Teunissen. Least-squares estimation of the integer gps ambiguities. In *Invited lecture, section IV theory and methodology, IAG general meeting, Beijing, China*, 1993.

[93] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The international journal of robotics research*, 23(7-8):693–716, 2004.

[94] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probalistic robotics. *Kybernetes*, 2006.

[95] Alberto Valero-Gomez, Javier Gómez, Santiago Garrido, and Luis Moreno. Fast marching methods in path planning. *IEEE Robotics amp amp amp Automation Magazine*, 20:111 – 120, 12 2013.

[96] Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric A Wan. The unscented particle filter. In *Advances in neural information processing systems*, pages 584–590, 2001.

[97] N.G. van Kampen. Remarks on Non-Markov Processes. *Brazilian Journal of Physics*, 28:90 – 96, 06 1998. ISSN 0103-9733. URL http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-97331998000200003&nrm=iso.

[98] Yunfeng Wang and Gregory S Chirikjian. A new potential field method for robot path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 977–982. IEEE, 2000.

[99] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

[100] Wei-Wen Kao. Integration of gps and dead-reckoning navigation systems. In *Vehicle Navigation and Information Systems Conference, 1991*, volume 2, pages 635–643, 1991.

[101] Håkon Jarvis Westergård. Fast marching and fast sweeping in optimal path planning. 2018.

[102] Zaw Win. On the windy postman problem on eulerian graphs. *Mathematical Programming*, 44(1-3):97–112, 1989.

[103] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, page 47–53, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919831. doi: 10.1145/280765.280773. URL https://doi.org/10.1145/280765.280773.

[104] Jinhui Yang, Xiaohu Shi, Maurizio Marchese, and Yanchun Liang. An ant colony optimization method for generalized tsp problem. *Progress in Natural Science*, 18(11):1417–1422, 2008.

[105] Yiming Ye and John K Tsotsos. Sensor planning for 3d object search. *Computer Vision and Image Understanding*, 73(2):145–168, 1999.

[106] A. Zelinsky, R.A. Jarvis, J. C. Byrne, and S. Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *In Proceedings of International Conference on Advanced Robotics*, pages 533–538, 1993.